AD-A106 684 STANFORD UNIV CA F/G 9/5
RESEARCH ON ALGORITHMS FOR ADAPTIVE ANTENNA ARRAYS.(U)
AUG 81 B WIDROW, W NEWMAN, R GOOCH, K DUVALL F30602-80-C-0046
UNCLASSIFIED RADC-TR-81-206 NL

1 of 2
AD
A106684

RADC-TR-81-[illegible]
Interim Report
August 1981

RESEARCH ON ALGORITHMS FOR ADAPTIVE ANTENNA ARRAYS

Stanford University

Bernard Widrow
W. Newman
R. Gooch
K. Duvall
D. Shur

[illegible]

[illegible]

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

19 **REPORT DOCUMENTATION PAGE**		READ INSTRUCTIONS BEFORE COMPLETING FORM
18 ~~1. REPORT NUMBER~~ RADC-TR-81-206	2. GOVT ACCESSION NO. AD-A106 684	~~3. RECIPIENT'S CATALOG NUMBER~~ Annual
6 4. TITLE (and Subtitle) RESEARCH ON ALGORITHMS FOR ADAPTIVE ANTENNA ARRAYS		9 ~~5. TYPE OF REPORT & PERIOD COVERED~~ Interim Report, Dec 79 – Dec 80
		~~6. PERFORMING ORG. REPORT NUMBER~~ N/A
10 7. AUTHOR(s) Bernard Widrow, W. Newman, R. Gooch, K. Duvall, D. Shur		15 ~~8. CONTRACT OR GRANT NUMBER(s)~~ F30602-80-C-0046 new
9. PERFORMING ORGANIZATION NAME AND ADDRESS Stanford University, Stanford CA 94305 332550		16 ~~10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS~~ 61102F 2305J805 17 J8
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (DCCL) Griffiss AFB NY 13441		11 ~~12. REPORT DATE~~ Aug~~ust 19~~81
		12 ~~13. NUMBER OF PAGES~~ 97
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Same		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: John T. Gamble (DCCL)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Adaptive Antennas
Adaptive Arrays
Adaptive Filters
LMS/Newton Algorithm
Null Steering

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The fundamental efficiency of adaptive algorithms is analyzed. It is found that noise in the adaptive weights increases with convergence speed. This causes loss in mean-square-error performance. Efficiency is considered from the point of view of misadjustment versus speed of convergence. A new version of the LMS algorithm based on Newton's method is anlayzed and shown to make maximally efficient use of real-time input data. The performance of this algorithm is not affected by eigenvalue

Item 20 (Cont'd)

disparity. Practical algorithms can be devised that closely approximate Newton's method. In certain cases, the steepest descent version of LMS performs as well as Newton's method.

The efficieny of adaptive algorithms with nonstationary input environments is analyzed where signals, jammers, and background noises can be of a transient and nonstationary nature. Such environments have been modeled in terms of moving paraboloidal mean-square-error performance functions. The bottom of the performance bowl can be assumed to move slowly, with a randomly correlated Markov character. Exponential time weighting, inherent in the LMS algorithm, can give optimal performance with the proper choice of the parameter u when the motion of the bottom of the bowl is first order Markov. With higher-order Markov activity, exponential time weighting is no longer optimal. Higher-order adaptive algorithms are devised for nonstationary input applications.

A new adaptive filtering method for broadband adaptive beamforming is described which uses both poles and zeros in the adaptive signal filtering paths from the antenna elements to the final array output. When directly adapting feedback coefficients, difficulties arise as the mean-square-error is not a quadratic function of the weights but is, in fact, multimodal. There are questions of process instability, convergence to local rather than global optima, and generally slow convergence that occurs with all of the known algorithms that have been proposed for adapting feedback filters. However, it is possible to simultaneously adapt feedforward and feedback filter coefficients by adapting feedforward filters only. The advantage of this approach comes from the quadratic nature of the mean-square-error function. This method does not have problems of instability and convergence to local optima, and it converges essentially as fast as the conventional zeros-only system. It has the disadvantage of not minimizing the true mean-square-error, but does minimize the mean square of a filtered version of the true error. The effects of not minimizing true mean-square-error are being analyzed. Some simulation experiments have been conducted to compare the new approach with conventional beamformers using an equal number of variable weights. Preliminary results show deeper and sharper nulling with the new approach than with the conventional approach.

Abstract

This annual interim report is organized in three parts:

Part I

The fundamental efficiency of adaptive algorithms is analyzed. It is found that noise in the adaptive weights increases with convergence speed. This causes loss in mean-square-error performance. Efficiency is considered from the point of view of misadjustment versus speed of convergence. A new version of the LMS algorithm based on Newton's method is analyzed and shown to make as efficient use of real-time input data as can be. The performance of this algorithm is not affected by eigenvalue disparity. Practical algorithms can be devised that closely approximate Newton's method. In certain cases, the steepest descent version of LMS performs as well as Newton's method.

Part II

This part analyzes the efficiency of adaptive algorithms with nonstationary input environments, i.e signals, jammers, and background noises can be of a transient and nonstationary nature. Such environments have been modeled in terms of moving paraboloidal mean-square-error performance functions. The bottom of the performance bowl can be assumed to move slowly, with a randomly correlated Markov character. Exponential time weighting, inherent in the LMS algorithm, can give optimal performance with the proper choice of the parameter μ when the motion of the bottom of the bowl is first order Markov. With higher-order Markov activity, exponential time weighting is no longer optimal. Higher order adaptive algorithms are being devised for nonstationary input applications.

Part III

This part introduces a new adaptive filtering method for broadband adaptive beamforming. It uses both poles and zeros in the adaptive signal filtering paths from the antenna elements to the final array output.

When directly adapting feedback coefficients, difficulties arise as the mean square error is not a quadratic function of the weights but is, in fact, multimodal. There are questions of process instability, hang up on local rather than global optima, and generally slow convergence that occurs with all of the known algorithms that have been proposed for adapting feedback filters.

Our methodology effectively permits simultaneous adaptation of feedforward and feedback filter coefficients by adapting feedforward filters only. The advantage of this approach comes from the quadratic nature of the mean square error function. Our method does not have problems of instability and convergence to local optima, and it converges essentially as fast as the conventional zeros-only system. It has the disadvantage of not minimizing the true mean-square-error, but does minimize the mean square of a filtered version of the true error. The effects of not minimizing true mean square error are being analyzed.

Preliminary experimental results comparing systems with equal numbers of variable weights, show deeper and sharper jammer nulling than would be possible with the conventional approach.

PART I

ADAPTIVE ALGORITHM EFFICIENCY

1.1 INTRODUCTION

The first part of this report deals with the efficiency of adaptive algorithms, and suggests a way to improve on the current techniques. We include a general discussion of the issues involved in evaluating and comparing various adaptive algorithms. Also, an optimal form of adaptive algorithm is introduced, called the LMS/Newton algorithm. This is a stochastic gradient descent algorithm based on Newton's Method, with a statistical performance which is as efficient in its data usage as possible. Eigenvalue spread does not affect the rate of convergence of LMS/Newton. The standard LMS algorithm and LMS/Newton algorithm are compared, and conditions are established when the two algorithms have the same average performance.

1.2 ALGORITHM EFFICIENCY

Two forms of the LMS adaptive algorithm will be discussed here, the "usual" algorithm based on the method of steepest descent [1-5], and an idealized algorithm based on Newton's method. These algorithms will be considered from the points of view of: (a) rate of convergence, (b) efficiency of statistical performance.

Speeding up a given adaptive process generally requires that the adaptive parameters (weights, etc.) take values based on averaging over less input data. The result is increased parameter noise and reduced average system performance. When using a specific algorithm, there is generally a tradeoff between speed of convergence and average statistical performance.

Two algorithms may be compared with each other when applied to the same adaptation task by adjusting their rates of convergence to cause the same effective parameter noise. As such, the more efficient algorithm converges faster. Effective parameter noise is that attribute of the noise that causes loss in system performance.

The steepest descent version of the LMS algorithm is

$$W_{j+1} = W_j + 2\mu\varepsilon_j X_j \quad . \tag{1.1}$$

The p^{th} mode of the mean square error learning curve has a time constant given by

$$\tau_{p_{mse}} = \frac{1}{4\mu\lambda_p} \quad . \tag{1.2}$$

It is seen that increasing the convergence factor μ speeds up the adaptive process by causing the adaptive time constants to be reduced. However, to insure stability in the mean, μ must be kept within the bounds

$$\frac{1}{\lambda_{max}} > \mu > 0 \quad , \tag{1.3}$$

where λ_{max} is the largest eigenvalue of the input correlation matrix R. After adaptive transients die out, noise in the weights causes, on the average, an increase in mean square error over the theoretical minimum mean square error. The misadjustment M has been defined [1-5] as the dimensionless ratio of the average excess mean square error to the minimum mean square error. For the steepest descent LMS algorithm,

$$M = \mu \; trace \; R = \frac{n}{4}\left(\frac{1}{\tau_{p_{mse}}}\right)_{ave} \tag{1.4}$$

Increasing μ speeds up the adaptive process but increases the misadjustment.

A "Newton's method" version of the LMS algorithm premultiplies the instantaneous gradient estimate $2\varepsilon_j X_j$ by the inverse of R. The algorithm is

$$W_{j+1} = W_j + 2\mu\lambda_{ave}{}_j R^{-1} X_j \quad . \tag{1.5}$$

The scaling constant λ_{ave} (the average of the eigenvalues) has been included for convenience. It can be shown that premultiplication by R^{-1} causes each adaptive step to be taken not along the maximum gradient but instead in the direction toward the bottom of a quadratic bowl. The effect is very much like application of steepest descent when all eigenvalues are equal. The eccentricity of the performance function is eliminated by Newton's method as specified by (1.5). Newton's method requires R^{-1} which is generally not available. An attempt to perform an algorithm like equation (1.5), only using an R^{-1} estimated from input data, has been reported by Griffiths and Mantey [6] and is summarized in section 2.2 of this report.

For now, we shall focus our attention on equation (1.5), realizing that such an algorithm, a true Newton's method version of LMS, is a mathematical idealization. It can be shown to have the following properties. Instead of there being a number of time constants of the mean square error learning curve equal to the number of weights (as with conventional LMS), there is a single time constant given by

$$\tau_{mse} = \frac{1}{4\mu\lambda_{ave}} \quad . \tag{1.6}$$

The misadjustment of algorithm (1.5) is given by

$$M = \mu \; trace \;\; R = \frac{n}{4}\left[\frac{1}{\tau_{mse}}\right] \quad . \tag{1.7}$$

The bounds on μ for convergence in the mean are

$$\frac{1}{\lambda_{ave}} > \mu > 0 \quad . \tag{1.8}$$

Comparing the two algorithms, we make their μ values equal in order to have equal misadjustments. Immediately we see that the stable range of μ for steepest descent is smaller than for Newton's method when there is eigenvalue disparity. Since these algorithms are generally operated with small μ to maintain small M, this is not necessarily disadvantageous for steepest descent. However, when the eigenvalue spread is extreme, steepest descent may be forced to operate with a very small value of μ in order to maintain stability. Under such circumstances, steepest descent would be stability bound rather than misadjustment bound.

With equal settings of μ, both algorithms have the same misadjustment. Under these circumstances, it is interesting to compare the Newton's-method time constant (1.6) with the steepest-descent time constants (1.2). It is clear that some of the steepest descent convergence modes are going to be faster while some are going to be slower than the single Newton's method mode. If we compare areas under the learning curves in order to compare "learning times" of single mode exponential curves with multimode curves (as is done in Fig. 1.1), it can be shown that when misadjustment bound, the learning time of steepest descent averaged over random initial conditions is identical to the learning time

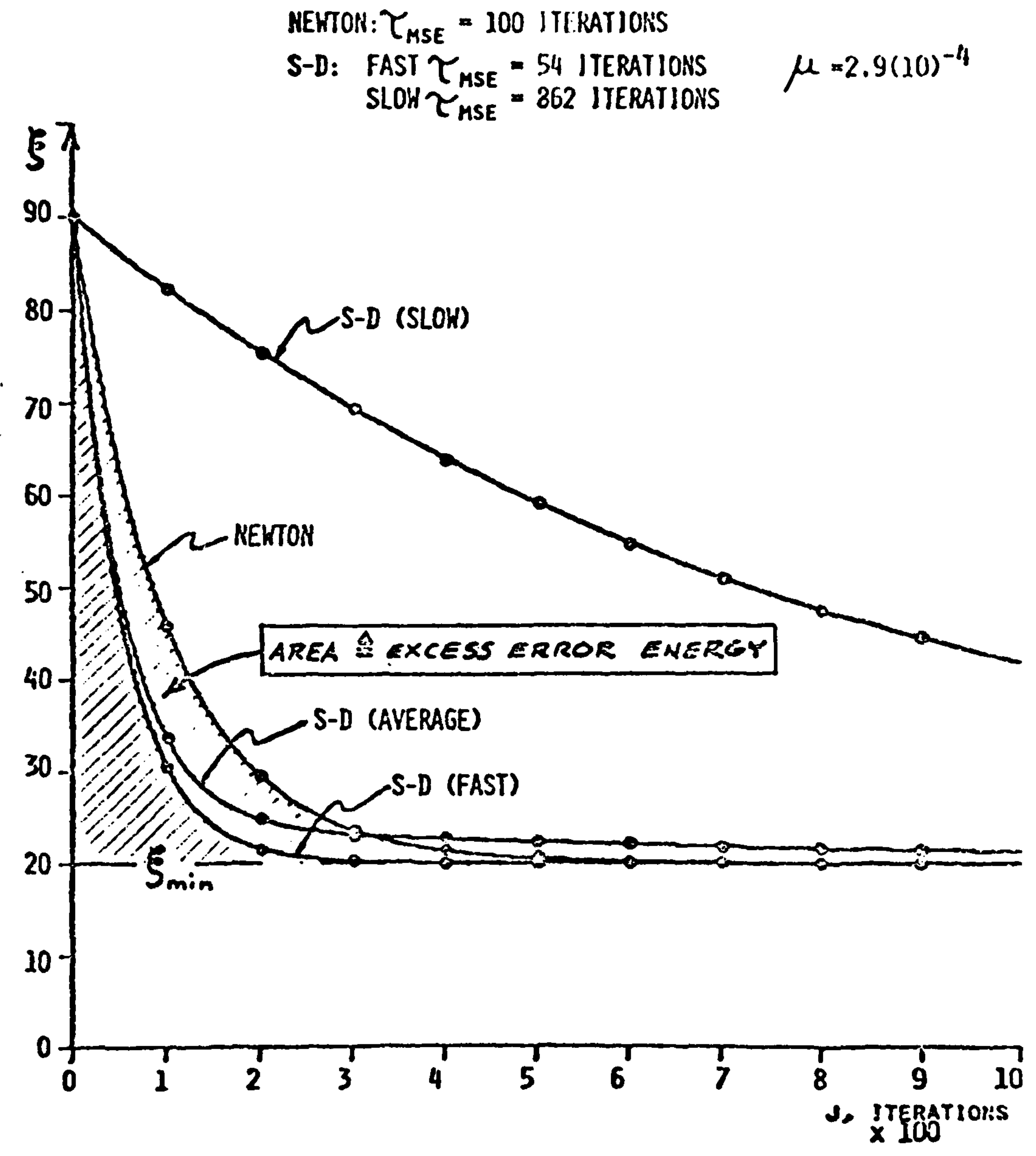


Fig. 1.1. Learning curves for Newton's method and the method of steepest descent (S-D).

of Newton's method. However, one should realize that the worst case learning time for steepest descent will be worse than that for Newton's method by a factor of $\lambda_{max}/\lambda_{ave}$.

The behavior of the steepest descent LMS algorithm has been analyzed in detail in [4] with a simple form of nonstationary input that results in the quadratic mean square error function undergoing a random vector displacement. The motion of the bottom of the bowl is first order Markov. Misadjustment results both from noise in the weights and from the weights dynamically lagging behind the bottom of the moving mean square error bowl. It is shown that the total misadjustment is minimized when the rate of adaptation is adjusted (by choice of μ) so that both components of misadjustment are equal. A similar analysis has been made for LMS Newton, and it has been found that the value of μ that optimizes steepest descent also optimizes Newton's method and that both algorithms yield the same misadjustment for the same μ. The conclusion is that if the steepest descent algorithm is misadjustment bound rather than stability bound, the conventional steepest descent approach gives identical performance in a statistical sense to Newton's method with simple nonstationary inputs.

The Newton's method version of the LMS algorithm is about as effecient as an algorithm can be, from the standpoint of statistical performance. For a given number of weights and for a given level of misadjustment, the number of data samples seen and consumed in the convergence process of LMS Newton is about as small as nature will permit. Justification for this comes from study of adaptive behavior when learning with a finite number of data samples.

It is shown in Appendix A of reference [4] that when training an n-weight adaptive system with N independent data vectors, the expected misadjustment is

$$M = \frac{n}{N} = \frac{\text{number of weights}}{\text{number of training samples}} . \tag{1.9}$$

This result was first reported without a complete proof in [1]. It is independent of algorithm, as long as the algorithm is *least squares*. A few simplifying assumptions were made to arrive at such an elegant and simple result. This formula has been tested extensively by computer simulation and has been found to be quite accurate for misadjustments of 25% or less.

Misadjustment formula (1.9) may be compared with that for LMS Newton (1.7). The comparison cannot be exact however, because (1.9) applies to learning with a finite block of data while (1.7) applies to a steady flow learning process. Actually, (1.9) applies to steady flow learning with a uniform moving-average window while (1.7) applies to steady flow learning with an exponential moving window. Reconsider equation (1.7). It can be written as

$$M = \frac{n}{4\tau_{mse}} \tag{1.10}$$

One can read this as "misadjustment of LMS Newton equals the number of weights divided by the number of training samples," if one considers that an exponential process essentially settles within four time constants and that any input that has occured more than four time constants ago would have negligible effect on the weights. We conclude that *LMS Newton has the misadjustment of a fundamental least squares process*. No adaptive process could have a lower misadjustment than (1.9) or (1.10). No more "information" (in the common English sense) can be squeezed from a given amount of data.

We now know that the Newton's method version of LMS is as fine and efficient an adaptive algorithm as can be. Unfortunately it cannot be implemented unless one perfectly knows R^{-1}. Without such knowledge, one can only approximate LMS Newton. In application to adaptive FIR digital filters, perhaps this or something approximating this is achieved by the adaptive lattice filter algorithms that have appeared during the last half dozen years or so. Whether or not this is true remains to be seen.

With extreme eigenvalue disparity, stability may be of limiting concern rather than misadjustment. To achieve maximum convergence speed with LMS steepest descent, μ would be set to $1/\lambda_{max}$, causing the slowest mode to have a time constant of $\lambda_{max}/4\lambda_{min}$ adaptations. Operating steepest descent at full speed, the misadjustment would be $M = traceR/\lambda_{max} > 1$. LMS Newton, doing the same job, could be pushed much faster. Maximum speed would be achieved by setting μ to half of its upper stable limit, i.e. $\mu = 1/2\lambda_{ave}$, giving theoretical (no gradient noise) convergence in one iteration. As such, its misadjustment would be $M = n/2$. A 100 weight filter, for example, would have a misadjustment of 50.

In most engineering applications, a misadjustment of 100% would be considered very high. The adaptive solution then gives twice as much mean square error as the Wiener solution, i.e. 3 dB more mean square error. Only when the minimum mean square error of the Wiener solution is zero or very small would high misadjustment be acceptable. Recall that misadjustment is normalized relative to minimum mean square error.

In most engineering applications, a misadjustment of about 10% would be satisfactory. As such, neither LMS/steepest descent nor LMS/Newton would be pushed anywhere near to the brink of instability. Speed of convergence would then be misadjustment limited rather than stability limited, regardless of eigenvalue spread.

It is safe to say that the steepest descent version of the LMS algorithm is the simplest, most widely used, and most widely understood of all adaptive algorithms. With all eigenvalues equal, its performance is identical to that of LMS Newton. With eigenvalue disparity, its average speed of convergence and statistical efficiency and performance with nonstationary inputs is identical to that of

LMS Newton, except that its worst case convergence rate is poorer.

When dealing with an input signal buried in noise, the eigenvalue spread is slight and one can be assured that LMS steepest descent will perform as well or better than any other algorithm. When the input is stationary, noise free or only slightly noisy, and when the input signal is narrow-band with a highly peaked spectrum, eigenvalue disparity could be large or extreme. Under these circumstances, opportunities will exist to better the worst-case performance of the steepest descent form of the LMS algorithm.

PART II

ADAPTIVE ALGORITHMS FOR NONSTATIONARY INPUTS

2.1 INTRODUCTION

The preceding discussion points out that comparisons between adaptive algorithms can be made on the basis of misadjustment and speed of convergence. Often the jammers to be nulled are of a time-varying nature. In this case, the concepts of misadjustment and speed of convergence have appropriate counterparts. For the time varying case, one can view the algorithm as being repeatedly restarted with only a few input data samples available to estimate the weight values required to null the changing jammer. As such, the speed of convergence of an algorithm relates to how well the algorithm 'tracks' or follows a change. If a jammer changes position or changes spectral character, then the optimal weights required to maintain the best possible jammer null must also change. Tracking error refers to the difference between the average value of the adapting weights and the optimal weights at each time instant. In a nonstationary environment, poor tracking implies that the ensemble mean of the adapting weight vector is far from the optimal weight vector which would produce the best possible null. Weight misadjustment is related to the noise in the adapting weights. If an adaptive algorithm is tuned to track a time-varying jammer, it will exhibit some amount of weight noise. In fact, tracking ability and weight noise are directly related. The better the tracking performance, the higher the weight noise. Ideally, one would like an algorithm which is capable of tracking a quickly varying jammer, yet show little weight noise. Part 2 of this report develops several algorithms which show improvements of this kind over conventional LMS.

This section begins with a discussion of the recursive least squares algorithm for adaptive antennas. The algorithm will be developed in an exact least squares sense, then a stochastic interpretation of the functioning of the algorithm will be given. The tradeoffs between misadjustment and tracking will be shown by actual simulations. A new algorithm will then be introduced as a method to improve the misadjustment/tracking tradeoff. Derivations of optimal strategies for use of data will then be made, and the circumstances under which the recursive least squares algorithm is optimal will be discussed. Comparisons between the new algorithm and current techniques will then be presented. The goal is to obtain optimal adaptive array performance with nonstationary inputs consisting of signal, noise, and jammers.

2.2 DERIVATION OF WRLS

The weighted recursive least squares algorithm (WRLS) analyzed here has been previously described by others [6]. This discussion is included in order to establish terminology and to introduce recursive algorithms that can be applied to adaptive antenna arrays.

Consider the Zahm beamformer of Fig 2.1. This antenna array is adapted so that the filtered output of the auxiliary elements $y(t)$ matches as closely as possible the output of the primary element $d(t)$. This results in an antenna array sensitivity pattern which is as omnidirectional as possible while maintaining nulls on the stronger incident signals. For the array of Fig. 2.1, the filtered output from each auxiliary element is

$$y_i(s,t) = \sum_{j=0}^{n} w_{i,j}(s)x_i(t-j) \ . \tag{2.1}$$

The weights are now assumed to be a function of the parameter s which will soon be used to determine an optimality criterion. Define the vectors,

$$W_i^T(t) = [w_{i,o}(t) \cdots w_{i,n}(t)]^T \ , \tag{2.2}$$

$$X_i^T(t) = [x_i(t), x_i(t-1), \cdots, x_i(t-n)]^T \ . \tag{2.3}$$

In vector notation, Eq. 2.1 becomes

$$y_i(s,t) = W_i^T(s)X_i(t) \ . \tag{2.4}$$

The summed output of the auxiliary elements is

$$\begin{aligned} Y(s,t) &= \sum_{i=1}^{m} y_i(s,t) \\ &= \sum_{i=1}^{m} W_i^T(s)X_i(t) \ . \end{aligned} \tag{2.5}$$

To simplify notation later, introduce the combined vectors,

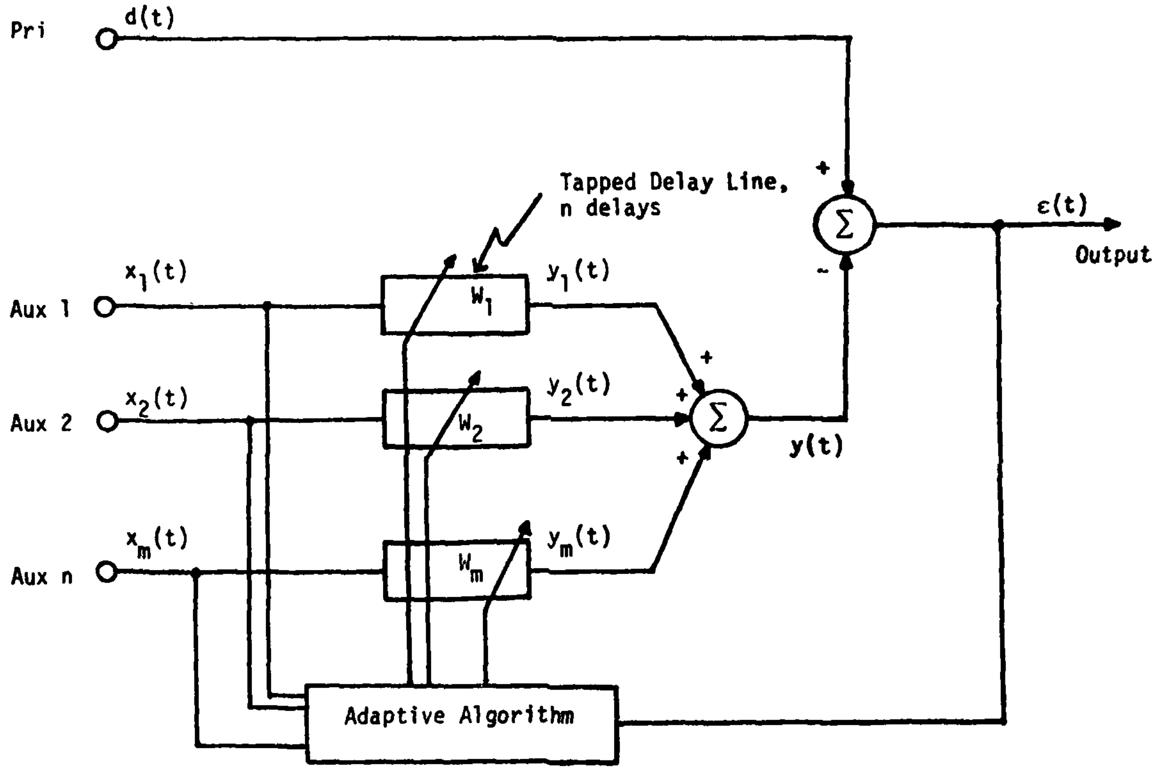


Figure 2.1. Zahm adaptive beamformer.

$$\mathbf{W}^T(t) = [W_1^T(t) \cdots W_m^T(t)] \tag{2.6}$$

$$\mathbf{X}^T(t) = [X_1^T(t) \cdots X_m^T(t)] \ . \tag{2.7}$$

so we get

$$Y(s,t) = \mathbf{W}^T(s)\mathbf{X}(t) \ . \tag{2.8}$$

The instantaneous error is given by,

$$\varepsilon(s,t) = d(t) - Y(s,t) \ . \tag{2.9a}$$

The error criterion to be used for adapting the weights is a weighted sum of squared errors,

$$\begin{aligned} \varepsilon'(t) &= \sum_{i=1}^{t} \alpha_i(t)\varepsilon(t,i)^2 \\ &= \sum_{i=1}^{t} \alpha_i(t)\left[d(i) - Y(t,i)\right]^2 \ . \end{aligned} \tag{2.9b}$$

The $\alpha_i(t)$ are weighting coefficients which are usually chosen to weight errors in the recent past more heavily than errors in the distant past. The derivative of $\varepsilon'(t)$ with respect to $\mathbf{W}(t)$ is

$$\begin{aligned} \frac{\partial \varepsilon'(t)}{\partial \mathbf{W}(t)} &= \frac{\partial}{\partial \mathbf{W}(t)} \sum_{i=1}^{t} \alpha_i(t)\left[d(i) - \mathbf{W}^T(t)\mathbf{X}(i)\right]^2 \\ &= \sum_{i=1}^{t} 2\alpha_i(t)\left[d(i) - \mathbf{W}^T(t)\mathbf{X}(i)\right]\mathbf{X}^T(i) \ . \end{aligned} \tag{2.10}$$

To minimize $\varepsilon'(t)$, this derivative is set to zero, giving

$$\begin{aligned} 0 &= \frac{\partial \varepsilon'(t)}{\partial \mathbf{W}(t)} \\ &= \sum_{i=1}^{t} 2\alpha_i(t)(d(i) - \mathbf{W}^T(t)\mathbf{X}(i))\mathbf{X}^T(i) \end{aligned} \tag{2.11}$$

or

$$\sum_{i=1}^{t} \alpha_i(t)\mathbf{X}(i)\mathbf{X}^T(i)\mathbf{W}(t) = \sum_{i=1}^{t} \alpha_i(t)d(i)\mathbf{X}(i) \ . \tag{2.12}$$

which reduces to

$$\mathbf{W}(t) = \left[\sum_{i=1}^{t} \alpha_i(t) X(i) X^T(i)\right]^{-1} \sum_{i=1}^{t} \alpha_i(t) d(i) X(i) \quad . \tag{2.13}$$

The optimal choice of $\mathbf{W}(t)$ is given by Eq. (2.13), this value will minimize the weighted error $\varepsilon'(t)$ of Eq. (2.9).

The basic problem associated with finding $\mathbf{W}(t)$ is that for each time step, an $(n+1)m$ by $(n+1)m$ matrix must be inverted. For notational convenience, define the matrix

$$P(t) = \left[\sum_{i=1}^{t} \alpha^i(t) X(i) X^T(i)\right]^{-1} \quad . \tag{2.14}$$

To find $\mathbf{W}(t+1)$, the matrix $P(t+1)$ must be found. Since

$$\begin{aligned} P(t+1) &= \left[\sum_{i=1}^{t+1} \alpha_i(t+1) X(i) X^T(i)\right]^{-1} \\ &= \left[\sum_{i=1}^{t} \alpha_i(t+1) X(i) X^T(i) + \alpha_{t+1}(t+1) X(t+1) X^T(t+1)\right]^{-1} \quad . \end{aligned} \tag{2.15}$$

Making certain assumptions about the form of the weighting $\alpha_i(t)$ results in a simple recursion in which $P(t+1)$ can be found from $P(t)$ in order $(m(n+1))^2$ operations. Specifically, choosing

$$\alpha_i(t) = \alpha^{t-i} \quad i=1, 2, \ldots, t \tag{2.16}$$

results in the widely used exponential weighting scheme, where the most recent error sum (2.9b), makes the greatest contribution to the running mean square error, and it has the greatest effect on the current value of $\mathbf{W}(t)$. Also, errors in the distant past make a relatively small contribution to $\varepsilon'(t)$, and therefore have a reduced effect in the determination of $\mathbf{W}(t)$. Figure 2.2 shows how the weighting given by (2.16) can be interpreted as a sliding exponential window on the mean square error. Assuming an exponential form for $\alpha_i(t)$ gives

$$\begin{aligned} \alpha_i(t+1) &= \begin{cases} \alpha^{t+1-i} & i = 1, 2, \ldots, t \\ 1 & i = t+1 \end{cases} \\ &= \begin{cases} \alpha \; \alpha_i(t) & i = 1, 2, \ldots, t \\ 1 & i = t+1 \end{cases} \end{aligned} \tag{2.17}$$

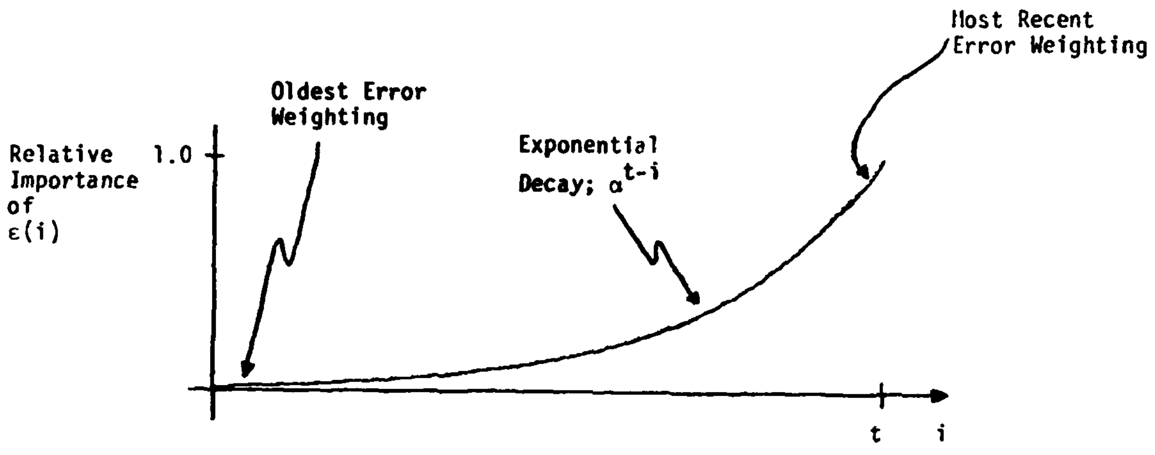


Figure 2.2. Exponential weighting of $\varepsilon(i)$.

so Eq. (2.15) reduces to

$$P(t+1) = \left[\alpha \sum_{i=1}^{t} \alpha_i(t)X(i)X^T(i) + X(t+1)X^T(t+1)\right]^{-1}$$
$$= \left[\alpha P^{-1}(t) + X(t+1)X^T(t+1)\right]^{-1} . \qquad (2.18)$$

The particular choice for a value of α will be made later. For now, assume that α is known. Equation 2.18 can be simplified using the matrix inversion lemma (Appendix A) to

$$P(t+1) = \frac{1}{\alpha}\left\{P(t) - \frac{P(t)X(t+1)X^T(t+1)P(t)}{\alpha + X^T(t+1)P(t)X(t+1)}\right\} . \qquad (2.19)$$

Equation (2.19) gives a recursive update for $P(t)$ which is far simpler than performing a direct matrix inversion at each step. Also, due to the form of $\alpha_i(t)$ the update for $W(t)$ becomes

$$W(t+1) = P(t+1)\left[\alpha\sum_{i=1}^{t} \alpha_i(t)d(i)X(i) + d(t+1)X(t+1)\right] . \qquad (2.20)$$

This expression can be combined with Eq. (2.19) to yield a recursive update for $W(t)$:

$$W(t+1) = W(t) + \frac{P(t)X(t+1)\left[d(t+1) - W^T(t)X(t+1)\right]}{\alpha + X^T(t+1)P(t)X(t+1)} . \qquad (2.21)$$

Appendix B contains all intermediate steps for deriving Eq. (2.21) from Eq. (2.20). Equations (2.19) and (2.21) are the update recursions for the WRLS algorithm.

2.3 MISADJUSTMENT AND TRACKING OF WRLS

The WRLS algorithm given by Eqs. (2.19) and (2.21) was implemented on a two-element Zahm beamformer shown in Fig. 2.3. The two element array had one incident time varying jammer, which generated a wavefront according to

$$W(t) = a_0(t)e(t) + a_1(t)e(t-1) \quad , \tag{2.22}$$

where $e(t)$ is a white-noise sequence. Note that if $a_0(t)$ and $a_1(t)$ are constants, the jammer is stationary. Allowing a time varying $a_0(t)$ and $a_1(t)$ makes the spectral character of the jammer change over time. The array spacing s and the jammer angle of incidence φ were chosen so that a time delay of one sampling period existed between the signals received by the primary and auxiliary elements. In the notation of Fig. 2.3, this is

$$x(t) = d(t-1) \quad . \tag{2.23}$$

Specializing the situation to that described by (2.23) allows an easy calculation of how the algorithm would optimally behave. This optimal behavior can be found as follows. The array output is given by

$$\begin{aligned} \varepsilon(t) &= d - w(t)x(t) \\ &= d(t) - w(t)d(t-1) \quad . \end{aligned} \tag{2.24}$$

The input is known to be generated by a process given by (2.22). An expression for $w(t)$ in terms of $a_0(t)$ and $a_1(t)$ can be found which represents the optimal choice of $w(t)$ to minimize $E(\varepsilon(t))$ for every t, since

$$\begin{aligned} E\left[\varepsilon^2(t)\right] &= E\left[(d(t) - w(t)d(t-1))^2\right] \\ &= E\left[W(t-n) - w(t)W(t-n-1)\right]^2 \\ &= E\left[a_0(t-n)e(t-n) + a_1(t-n)e(t-n-1)\right] \\ &\quad - w(t)\left[a_0(t-n-1)e(t-n-1) + a_1(t-n-1)e(t-n-2)\right]^2 \\ &= \left[a_0^2(t-n) + a_1^2(t-n) - 2w(t)(a_1(t-n)a_0(t-n-1)\right] \\ &\quad + w^2(t)\left[a_0^2(t-n-1) + a_1^2(t-n-1)\right] \end{aligned} \tag{2.25}$$

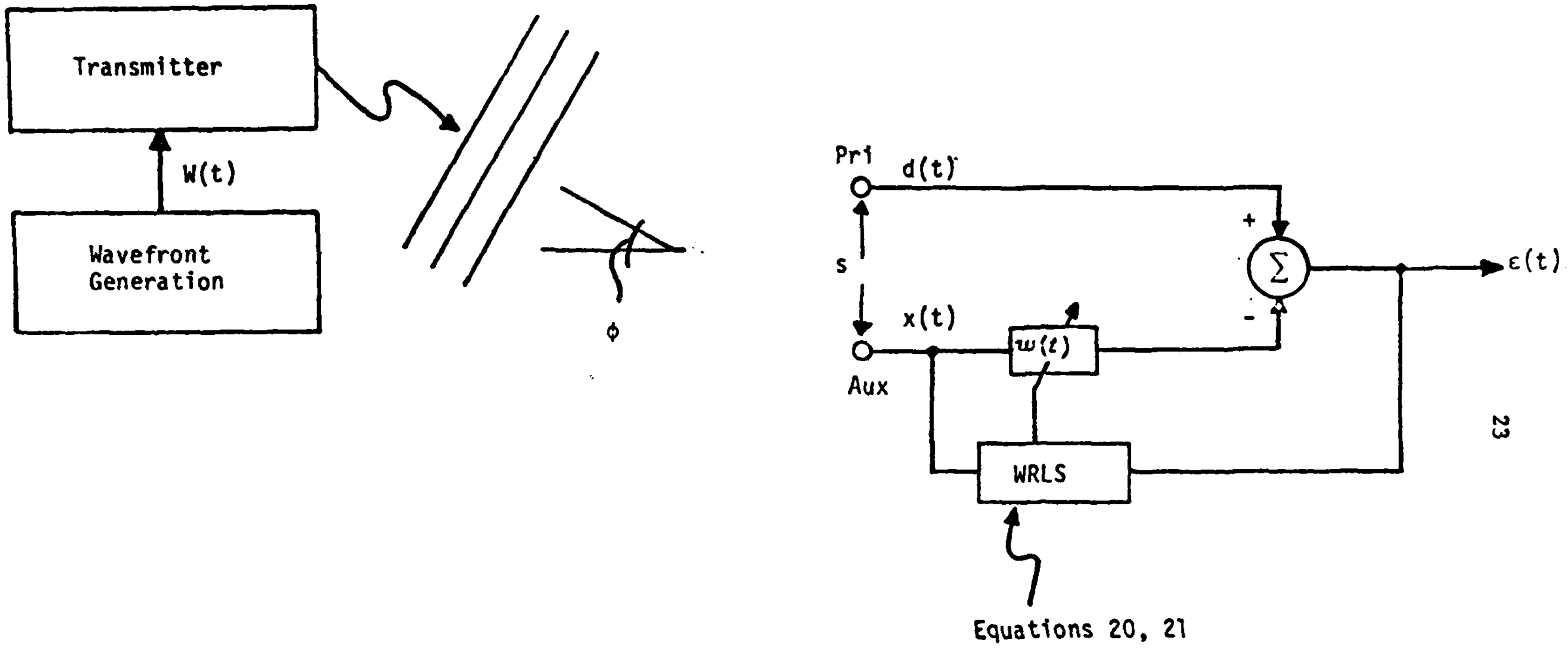


Figure 2.3. A simple, one weight Zahm adaptive array.

Taking derivatives and equating to zero gives the optimal $w(t)$, denoted by $w^*(t)$, as

$$0 = -2\left[a_1(t-n)a_0(t-n-1)\right] + 2w^*(t)\left[a_0^2(t-n-1) + a_1^2(t-n-1)\right] \quad , (2.26)$$

or

$$w^*(t) = \frac{a_1(t-n)a_0(t-n-1)}{a_0^2(t-n-1) + a_1^2(t-n-1)} \quad . \qquad (2.27)$$

This formula for the optimal weight value requires complete statistical knowledge about the jamming signal and is used only as a benchmark by which to compare practical adaptive antenna algorithms such as WRLS. The WRLS algorithm (2.19,2.21) was used to estimate $w(t)$ and plots are shown comparing $w(t)$ to $w^*(t)$ (Fig. 2.4).

The plots in Fig. 2.4 illustrate the tradeoff between weight tracking speed and weight estimate variance. Figure 2.4a is a plot of $w^*(t)$, the optimal weight trajectory. Figures 2.4b,c,d are plots of the adapting $w(t)$ for progressively larger values of α, the decay rate factor. A graph of α^{t-t} is shown with each plot to give some visual idea of the relative error weightings. Note that decreasing α (placing more importance on new data) leads to better average tracking of the true parameter, but the adapting weight variance (misadjustment) is quite large. Conversely, increasing α (corresponding to a decrease in the importance placed on new data) the algorithm lags behind and tracks the optimal weight poorly, but the adapting weight value has a low variance. The adapting weight of Fig. 2.4b is considered to have high miadjustment, while the adaptive weight of Fig. 2.4d has poor tracking qualities. In typical applications of this algorithm, a choice of α is based on the tradeoff between these two properties. Such a choice is shown in Fig. 2.4c, where both low misadjustment and good tracking properties are obtained. In the sections to follow, we develop a new class of algorithms

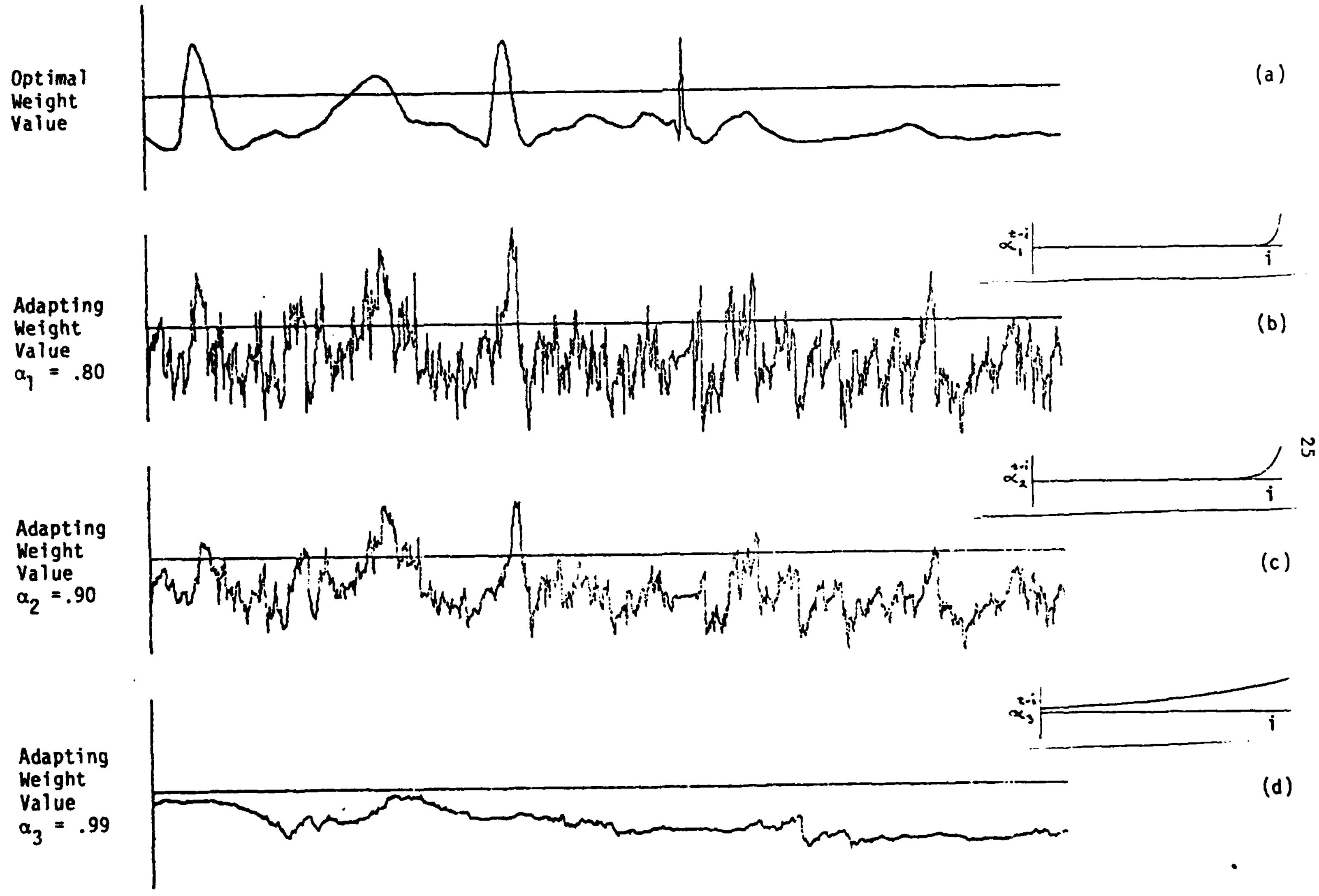


Figure 2.4. Weight variance and tracking for different choices of α.

which perform more efficiently in the misadjustment-versus-tracking sense described above.

2.4 STOCHASTIC INTERPRETATION OF WRLS

Note that the above derivation of the iterative recursions relied on a deterministic weighted least squares error criterion. Equations (2.19) and (2.21) find a set of weights $W_i(t)$ which are optimal for minimizing $\varepsilon'(t)$, based on the data sequence observed. From a stochastic point of view, this algorithm can be viewed as an approximate realization of the ideal LMS/Newton algorithm. A stochastic interpretation can be given for many of the quantities used in finding the weight updates (2.19,2.21). The weighted mean squared error criterion can be used as a performance measure, given by

$$E[\varepsilon'(t)] = E\left[\sum_{i=1}^{t} \alpha_i(t)(d(i) - Y(t,i))^2\right] . \tag{2.28}$$

Proceeding in exactly the same way as before, we minimize this expected mean squared error criterion by taking a derivative,

$$\begin{aligned} \frac{\partial E(\varepsilon'(t))}{\partial \underline{W}(t)} &= \frac{\partial}{\partial \underline{W}(t)} E\left[\sum_{i=1}^{t} \alpha_i(t)(d(i) - \underline{W}^T(t)\underline{X}(i))^2\right] \\ &= E\left[\sum_{i=1}^{t} 2\alpha_i(t)(d(i) - \underline{W}^T(t)\underline{X}(i))\underline{X}^T(i)\right] . \end{aligned} \tag{2.29}$$

Setting this derivative to zero gives,

$$E\left[\sum_{i=1}^{t} \alpha_i(t)\underline{X}(i)\underline{X}^T(i)\right]\underline{W}(t) = E\left[\sum_{i=1}^{t} \alpha_i(t)d(i)\underline{X}(i)\right] . \tag{2.30}$$

So,

$$\begin{aligned} \underline{W}^*(t) &= \left[E \sum_{i=1}^{t} \alpha_i(t)\underline{X}(i)\underline{X}^T(i)\right]^{-1} \left[E \sum_{i=1}^{t} \alpha_i(t)d(i)\underline{X}(i)\right] \\ &= \left[\left[\sum_{i=1}^{t} \alpha_i(t)\right]^{-1} E \sum_{i=1}^{t} \alpha_i(t)\underline{X}(i)\underline{X}^T(i)\right]^{-1} \\ &\quad \cdot \left[\left[\sum_{i=1}^{t} \alpha_i(t)\right]^{-1} \sum_{i=1}^{t} \alpha_i(t)d(i)\underline{X}(i)\right] . \end{aligned} \tag{2.31}$$

If the $X(t)$ sequence is stationary, the term contained in the inverse brackets of

(2.31) converges to an estimate of the autocorrelation matrix R_{xx},

$$\left[\sum_{i=1}^{t} \alpha_i(t)\right]^{-1} E \sum_{i=1}^{t} \alpha_i(t)X(i)X^T(i) = \left[\sum_{i=1}^{t} \alpha_i(t)\right]^{-1} \sum_{i=1}^{t} \alpha_i(t)E[X(i)X^T(i)]$$
$$= R_{xx}\left[\sum_{i=1}^{t} \alpha_i(t)\right]^{-1} \left[\sum_{i=1}^{t} \alpha_i(t)\right]$$
$$= R_{xx} \quad . \tag{2.32}$$

Similarly the last term in (2.31) converges to an estimate of the cross-correlation matrix P_{xd},

$$\left[\sum_{i=1}^{t} \alpha_i(t)\right]^{-1} E \sum_{i=1}^{t} \alpha_i(t)d(i)X(i) = \left[\sum_{i=1}^{t} \alpha_i(t)\right]^{-1} \left[\sum_{i=1}^{t} \alpha_i(t)E(d(i)X(i)\right]$$
$$= P_{xd}\left[\sum_{i=1}^{t} \alpha_i(t)\right]^{-1} \left[\sum_{i=1}^{t} \alpha_i(t)\right]$$
$$= P_{xd} \quad . \tag{2.33}$$

Hence, in the stochastic sense the weight vector is given by

$$W(t) = \hat{R}_{xx}^{-1}(t)\hat{P}_{xd}(t) \tag{2.34}$$

where $\hat{R}_{xx}^{-1}(t)$ and $\hat{P}_{xd}(t)$ converge in the mean to the true R_{xx} and P_{xd} values if the input signals are stationary. This is consistent with Wiener filter theory.

In the case that the input signals are not stationary but are time-varying, the quantities $\hat{R}_{xx}^{-1}(t)$ and $\hat{P}_{xd}^{-1}(t)$ are sample mean estimates of the true time-varying covariance matrix and cross-covariance vector. This is an important point to be remembered for the discussion that follows. *In situations where the input signals are non-stationary, the true covariance quantities R_{xx} and P_{xd} will be functions of time denoted by $R_{xx}(t)$ and $P_{xd}(t)$.*

Now, components of the iterative relations (2.19) and (2.21) can be compared with those of the LMS/Newton algorithm. Normalizing (2.35) gives the following recursive relations:

$$\mathbf{W}(t+1) = \mathbf{W}(t) + \frac{1}{\alpha + X^T(t+1)P(t)X(t+1)} P(t)X(t+1)(d(t+1) - \mathbf{W}^T(t)X(t+1))$$
$$= \mathbf{W}(t) + \frac{1}{K\alpha + X^T(t+1)KP(t)X(t+1)} KP(t)X(t+1)(d(t+1)$$
$$- \mathbf{W}^T(t)X(t+1)) \tag{2.35}$$

where the normalized constant K is chosen to be

$$K = \sum_{i=1}^{t} \alpha_i(t) \quad . \tag{2.36}$$

For the case of exponential $\alpha_i(t)$ this reduces to

$$K = \sum_{i=1}^{t} \alpha^{t-i}$$
$$= \frac{1-\alpha^t}{1-\alpha} \tag{2.37}$$

This choice of K gives

$$K\,P(t) = \hat{R}_{xx}^{-1}(t) \tag{2.38}$$

Hence the algorithm (2.19) and (2.21) constitute a form of LMS/Newton algorithm, with an adaptation coefficient which is time variable;

$$\mu(t) = \frac{1}{K\alpha + X^T(t+1)\,R_{xx}^{-1}(t)X(t+1)}$$
$$= \frac{1}{\frac{\alpha(1-\alpha^t)}{1-\alpha} + X^T(t+1)\hat{R}_{xx}^{-1}(t)X(t+1)} \tag{2.39}$$

So this gives the $W(t)$ adaptive update algorithm as

$$\mathbf{W}(t+1) = \mathbf{W}(t) + \mu(t)\hat{R}_{xx}^{-1}(t)X(t+1)\left[d(t+1) - \mathbf{W}^T(t)X(t+1)\right]. \tag{2.40}$$

which is quite similar to the LMS/Newton form given in Part I. An interpretation for the action of the $\mu(t)$ can be found by looking at the $X^T(t+1)\hat{R}_{xx}^{-1}(t)X(t+1)$ term. If $X(t)$ is assumed to be a zero mean gaussian process, then the probability of a given $X(t)$ occurring is

$$P[X(t) = x] = \frac{1}{(2\pi)^{n/2}(\det R_{xx})^{1/2}} e^{-X^T(t) R_{xx}^{-1} X(t)} \quad . \tag{2.41}$$

Where R_{xx} is the covariance matrix of the multivariate density. Since R_{xx} is a positive definite matrix, excursions of $X(t)$ from the mean are decreasingly probable. Also, since $X^T R_{xx}^{-1} X$ increases for less probable $X(t)$, $\mu(t)$ is decreased. Hence $\mu(t)$ can be thought of as automatically scaling the step size by the relative probability associated with a particular value of $X(t)$.

As is clear from the previous development, the WRLS and LMS/Newton algorithms both perform an exponential weighting of incoming data. Single mode exponential weighting is refered to here as a first order process. This terminology comes from the form of the recursion used to generate exponential weighting. For this reason WRLS and LMS/Newton are called first order adaptive algorithms. In the next section we explore the possibilities of higher order data weighting schemes and the resulting higher order adaptive algorithms.

2.5 ALTERNATIVES TO EXPONENTIAL DATA WEIGHTING

This section develops the intuitive justification for why nonexponential windowing coefficients, $\alpha_i(t)$, are desirable. A particular choice of the $\alpha_i(t)$ can be made by defining a suitable performance measure and by assuming some knowledge of the nonstationary character of input signals, jamming interferences, and noise.

An exponential form for the windowing coefficients was assumed in section 2.2 in the development of the WRLS algorithm. The exponential form leads to a simple, recursive way of estimating $P(t)$, the inverse sample covariance matrix. The unnormalized expression for $P^{-1}(t)$ is

$$\begin{aligned} P^{-1}(t-1) &= \hat{R}_{xx}(t+1) \\ &= \sum_{i=1}^{t} \alpha_i(t)X(i)X^T(i) \\ &= \sum_{i=1}^{t} \alpha^{t-i}\, X(i)X^T(i) \end{aligned} \tag{2.42}$$

This expression can be cast into a first-order recursive difference equation form as

$$\hat{R}_{xx}(t+1) = \alpha\hat{R}_{xx}(t) + X(t)X^T(t) \quad . \tag{2.43}$$

A more general class of estimators for $\hat{R}_{xx}(t)$ can be obtained by allowing higher order autoregressive moving average (ARMA) difference equations of the form,

$$\begin{aligned} \hat{R}_{xx}(t+1) = {} & a_0\hat{R}_{xx}(t) + \cdots + a_n\hat{R}_{xx}(t-n) + b_0X(t)X^T(t) \\ & + \cdots + b_mX(t-m)X^T(t-m) \quad . \end{aligned} \tag{2.44}$$

This ARMA representation of the estimator has time invariant coefficients and can therefore be viewed as a linear, time invariant filter acting on $X(t)X^T(t)$, to produce $\hat{R}_{xx}(t)$. The z-transform of the process given by (2.44) is

$$A(z)\hat{R}_{xx}(z) = B(z)X(z)X^T(z) \tag{2.45}$$

where $A(z)$ and $B(z)$ are polynomials of the form

$$A(z) = 1 - a_1 z^{-1} - a_2 z^{-2} - \cdots - a_n z^{-n} \tag{2.46}$$

$$B(z) = b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_m z^{-m} \quad , \tag{2.47}$$

and z^{-1} represents a unit delay. Equation (2.45) can be rewritten as

$$\hat{R}_{xx}(z) = \frac{B(z)}{A(z)} X(z) X^T(z) \quad . \tag{2.48}$$

The polynomial ratio $B(z)/A(z)$ represents the z-transform of a filter with input $X(z)X^T$ and output $\hat{R}_{xx}(z)$. Figure 2.5 illustrates the idea. The $B(z)$ polynomial possesses m roots. These roots are commonly referred to as 'zeroes' of the filter. Similarly the $A(z)$ polynomial has n roots called the 'poles' of the filter. These names come from the effect a root of the polynomial has on the filter transfer function,

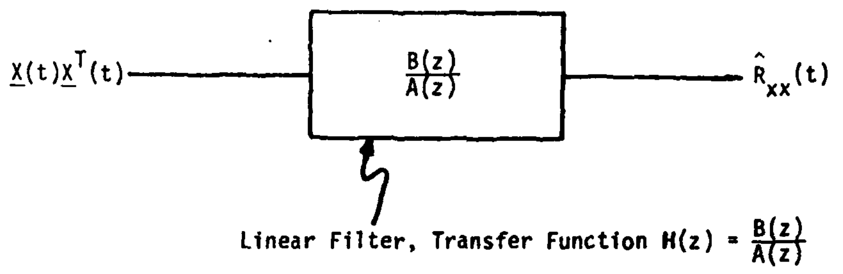
$$H(z) = \frac{B(z)}{A(z)} \quad . \tag{2.49}$$

A root of $B(z)$ will cause a zero in $H(z)$, a root in $A(z)$ will cause an infinite value (or pole) in the value of $H(z)$. This z-transform notation has been introduced so that the frequency response of a filter may be easily understood. The free variable z is a complex number, which when given the value,

$$z = e^{j\omega T} \tag{2.50}$$

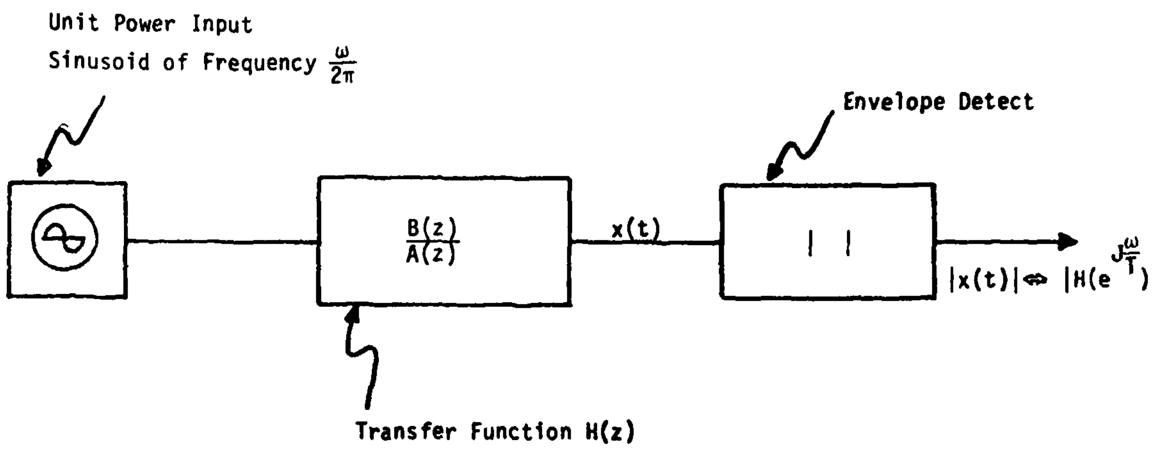
and substituted into (2.49) gives a complex number for $H(z)$. The magnitude of this number represents the magnitude of the response of the filter to a sinusoidal input at frequency $f = \frac{\omega}{2\pi}$. Figure 2.6 shows this idea.

A frequency domain interpretation of (2.48) lends valuable insight into the significance of allowing a higher order estimator. The z-transform of the first order estimator of Eq. (2.43) is



$\underline{X}(t)\underline{X}^T(t)$
$\frac{B(z)}{A(z)}$
$\hat{R}_{xx}(t)$
Linear Filter, Transfer Function $H(z) = \frac{B(z)}{A(z)}$

Figure 2.5. Representation of the general estimator as a linear filter.



Unit Power Input
Sinusoid of Frequency $\frac{\omega}{2\pi}$
Envelope Detect
$\frac{B(z)}{A(z)}$
$x(t)$
$| \ |$
$|x(t)| \Leftrightarrow |H(e^{j\frac{\omega}{T}})$
Transfer Function $H(z)$

Figure 2.6. Evaluation of frequency response of a linear filter.

$$(1-\alpha z^{-1})\hat{R}_{xx}(z) = X(z)X^T(z) \tag{2.51}$$

or

$$\hat{R}_{xx}(z) = \left[\frac{1}{1-\alpha z^{-1}}\right]X(z)X^T(z) \quad . \tag{2.52}$$

This is the same transform that was performed to arrive at (2.48). This is shown in Fig. 2.7. The form of (2.52) tells us that the exponential window acts as a single pole filter. This single pole estimator is a low-pass filter with cutoff α. Intuitively, this means the estimator passes low frequency (slowly varying) components of the $X(t)X^T(t)$ sequence, while rejecting high frequency (quickly varying) components of $X(t)X^T(t)$. A plot of the response of the filter described by (2.52) as a function of frequency is shown in Fig. 2.8. Note that the general estimator of Eq. (2.48) allows many poles *and* zeroes. Because of this, proper choice of the a_i and b_i coefficients in (2.46) and (2.47) will place dips and peaks in the frequency response of the filter. In the case of the single pole estimator, the only choice to be made was *where* the frequency response starts to drop. The more general formulation with its increased degrees of freedom allows a fairly arbitrary frequency response.

The usefulness of the more general formulation can be pointed out with a simple example. Recall that in situations where the jammers are nonstationary, the covariance matrix will be dependent on time. Suppose it is known that $R_{xx}(t)$ as a function of time, has a certain spectrum. A component of the $R_{xx}(t)$ matrix will typically have a spectrum as shown in Fig. 2.9a. A finite time average of the same component of $X(t)X^T(t)$ will usually have a spectrum like Fig. 2.9b. Generally, simple averaging or exponential averaging (as in section 2.2) of $X(t)X^T(t)$ will not give a spectrum which matches the true $R_{xx}(t)$ spectrum. This discrepancy arises because for non-stationary situations, the time averages used to estimate $R_{xx}(t)$ are not the same as the ensemble averages which

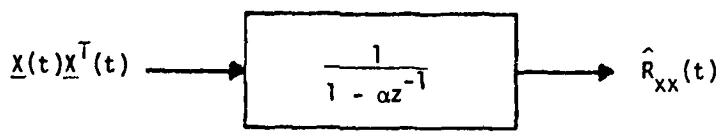

$\underline{X}(t)\underline{X}^T(t)$
$\frac{1}{1 - \alpha z^{-1}}$
$\hat{R}_{XX}(t)$

Figure 2.7. Z-transform representation of first order estimator.

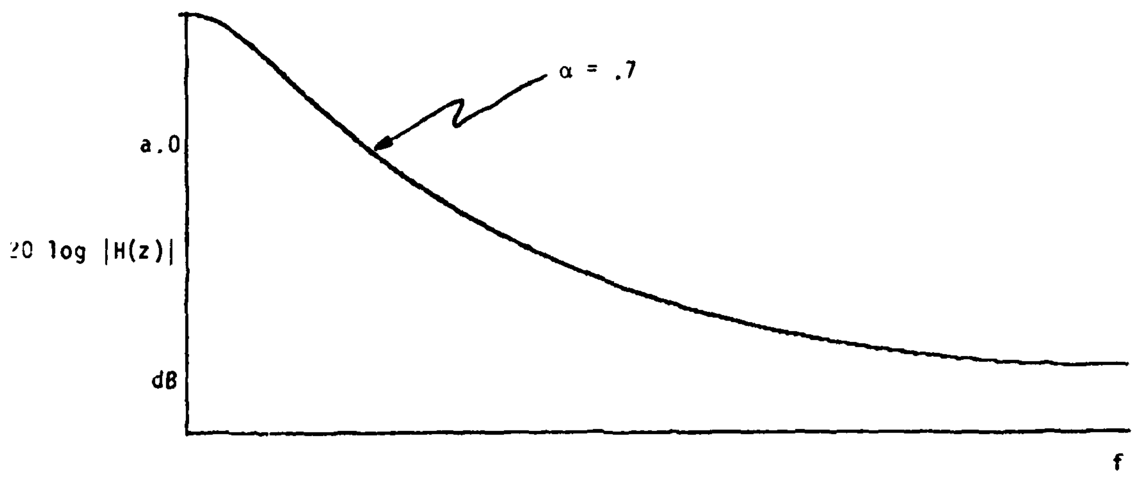
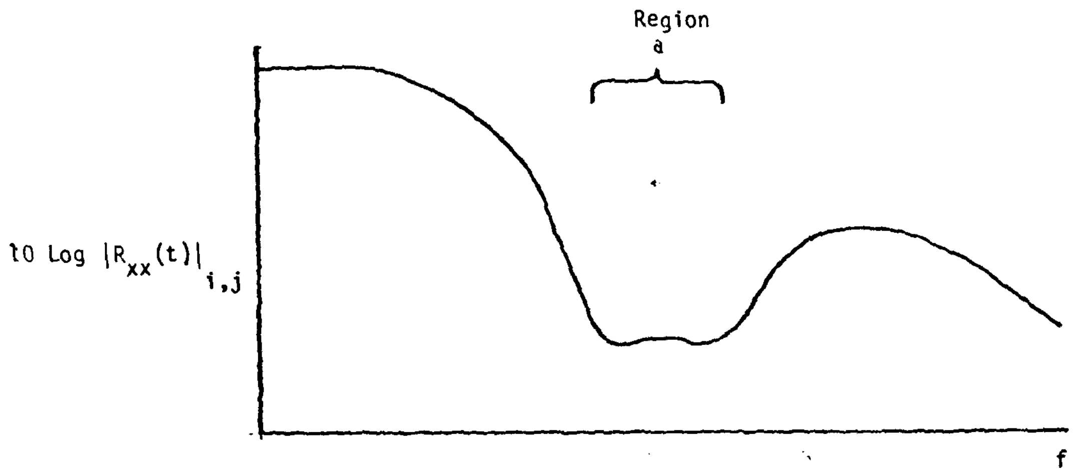


Figure 2.8. Frequency response of first order estimator.



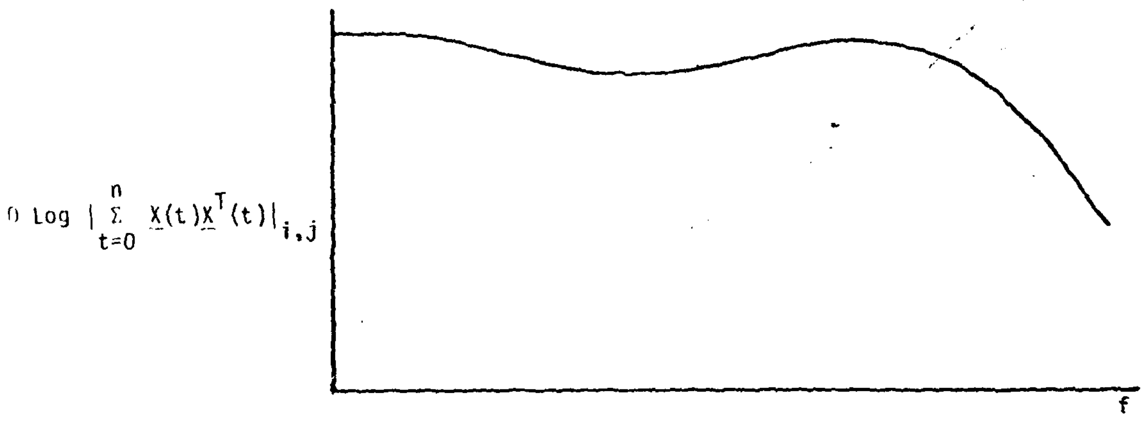


Figure 2.9. Spectra of true $R_{xx}(t)$ component and corresponding component of averaged $X(t)X^T(t)$ sequence.

constitute the true $R_{xx}(t)$. Clearly, something better than a simple average must be used. Intuitively, the estimator acting on the data of Fig. 2.9b should attempt to notch out those components with small magnitude in Fig. 2.9a. A general filter such as the one in Fig. 2.5 could do this easily, while the single pole filter of Fig. 2.7 would be a compromise at best. Figure 2.10 shows this pictorially. This is very similar to the matched filter concepts used in signal detection theory.

For this preliminary investigation, an all-zero filter approximation to the pole-zero form of (2.48) will be made. As shown above, the first-order estimator for $R_{xx}(t)$ had two expressions: a single pole (autoregressive) form, Eq. (2.43), and a multiple zero (moving average) form, Eq. (2.42). Equation (2.43) is called a parsimonious representatiion of the estimator, since it requires only one parameter (α) to describe it. Since practical considerations require a fixed memory size, Eq. (2.42) cannot be implemented directly. A good approximation can be made by choosing a sufficiently large m so that $\alpha^m = 0$, which gives

$$\begin{aligned}\hat{R}_{xx}(t) &= \sum_{i=1}^{t} \alpha^{t-i} X(i)X^T(i) \\ &= \sum_{i=t-m}^{t} \alpha^{t-i} X(i)X^T(i) \quad . \qquad (2.53)\end{aligned}$$

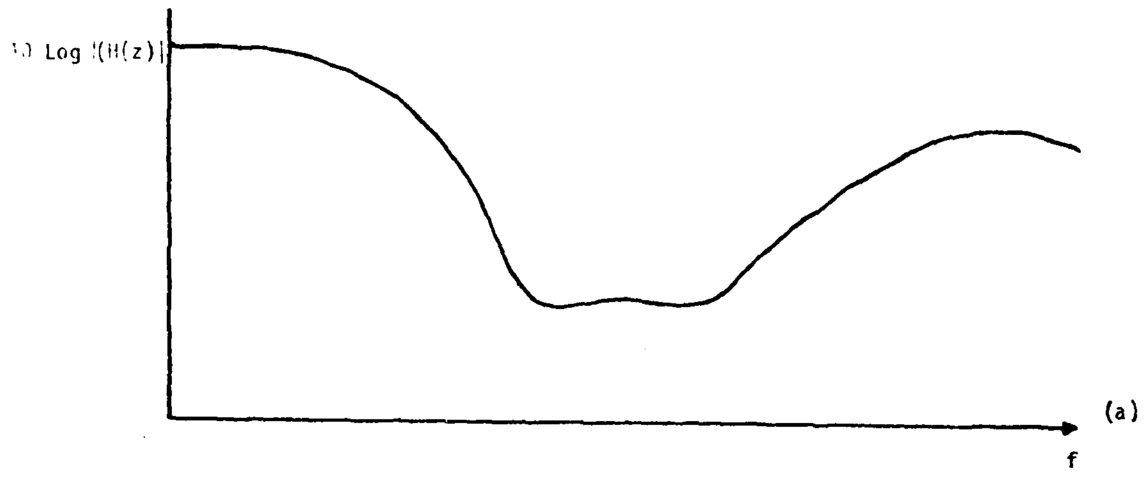
A development for the general autoregressive model demonstrates that if

$$\hat{R}_{xx}(t) = a_0\hat{R}_{xx}(t) + \cdots + a_n\hat{R}_{xx}(t-n) + X(t)X^T(t) \quad , \qquad (2.54)$$

it can also be estimated by a moving average process,

$$\hat{R}_{xx}(t) = \sum_{i=t-l}^{t} \alpha_i'(t)X(i)X^T(i) \quad . \qquad (2.55)$$

The representation is perfectly accurate if $l = t-1$, and is usually quite good if l is taken to be large but fixed. A proof of this result can be found in [10]. The result of these steps is that the estimator for $R_{xx}(t)$ (2.44) can be represented



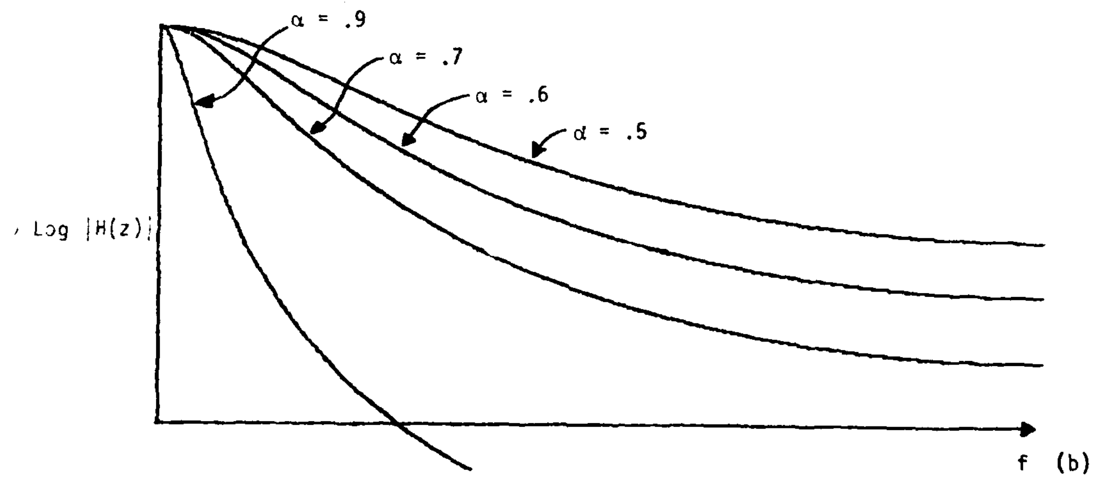


Figure 2.10. (a) Desired spectral character of $X(t)X^T(t)$ filter.
(b) Single pole approximation--chosen by user for best tracking/misadjustment combination.

by the simple moving average process

$$\hat{R}_{xx}(t) = \sum_{i=t-l}^{t} \alpha_i(t) X(i) X^T(i) \quad . \tag{2.56}$$

The cost of formulating the problem this way is one of a large memory requirement (more than is absolutely necessary); the advantage is a form for the estimator which is readily calculated.

Since Eq. (2.44) had constant coefficients, the $\alpha_i(t)$ coefficients will depend only on the index $t-i$. This result can be found in [10] So define

$$c(t-i) = \alpha_i(t) \tag{2.57}$$

which gives

$$\hat{R}_{xx}(t) = \sum_{i=t-l}^{t} c(t-i) X(i) X^T(i) \quad . \tag{2.58}$$

Equation (2.58) represents $\hat{R}_{xx}(t)$ as a convolution of the sequence $c(i)$ and $X(i)X^T(i)$. In this context, the $c(i)$ have an interpretation as the impulse response to the linear filter of Fig. 2.5 and can be drawn as in Fig. 2.11.

Up to this point, the concept of windowing has been applied only to estimation of $R_{xx}(t)$, but $P_{xd}(t)$ must also be estimated by the same window,

$$\hat{P}_{xd}(t) = \sum_{i=t-l}^{t} c(t-i) X(i) d(i) \quad . \tag{2.59}$$

Note that because arbitrary c(i) coefficients are allowed, the efficient (order $((n+1)m)^2$) update will not work. Hence $R_{xx}(t)$ must be found using equation 2.58, then inverted at each time step. A single inversion of $R_{xx}(t)$ requires an order of $((n+1)m)^3$ operations, which is not computationally acceptable. Computational issues will not be considered presently, since the point of this investigation is to determine if allowing arbitrary windowing coefficients $c(i)$ results in an improvement of the algorithm's performance. In subsequent investigations, the iterative inverse algorithm [11] will be used as a means of alleviating this

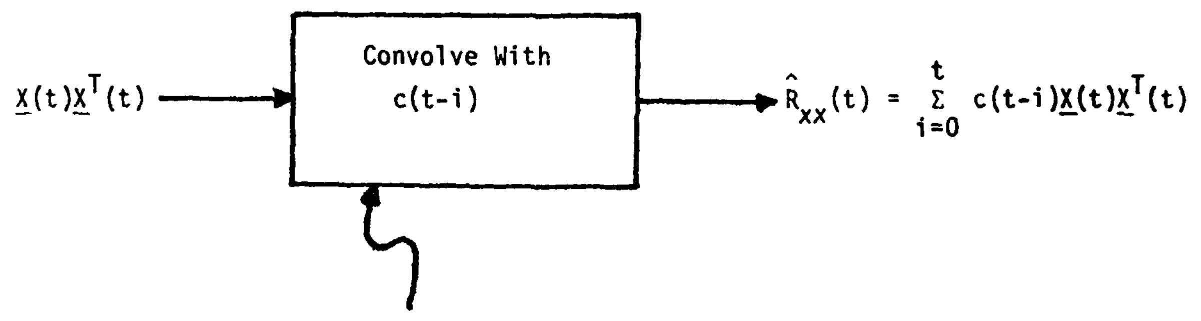

$\underline{X}(t)\underline{X}^T(t)$
Convolve With
c(t-i)
$\hat{R}_{xx}(t) = \sum_{i=0}^{t} c(t-i)\underline{X}(t)\underline{X}^T(t)$
Impulse Response of Filter of Figure 2.5.

Figure 2.11. Realization of Filter of Figure 2.5.

computational burden.

2.6 AN OPTIMAL WINDOW CRITERION

The adaptive algorithm given in section 2.2 (Eqs. 2.19, 2.21) forms estimates $R_{xx}(t)$ and $P_{xd}(t)$. Using these estimates the algorithm finds a weight vector $W(t)$ which will place a null on the jammers present. Section 2.4 discussed the fact that when the jammers are time varying, R_{xx} and P_{xd} become time varying quantities as well. Therefore, the problem is reduced to estimating time varying covariance quantities, $R_{xx}(t)$ and $P_{xd}(t)$, as accurately as possible, to null the jammer. It is desired to form this null quickly, while simultaneously maintaining an acceptable level of weight variance. Weight variance is undesirable since it may lead to signal cancellation [12]. Generally, the time varying jammer parameters are assumed to be stochastic processes. For this development, a certain subset of the jammer parameters Θ_t, will be allowed to change in time, and their values will be found as the output of a filter driven by white noise (see Fig. 2.12). Specifically, the jammer will have time varying frequency characteristics as shown in Fig. 2.13.

$R_{xx}(t)$ and $P_{xd}(t)$ will be found as weighted sums of past data, as shown in Eqs. (2.58)and (2.59), of section 2.5. With the above formulation, the idea is to now choose the window coefficients $c(i)$ so that $\hat{R}_{xx}(t)$ is in some sense a best estimate of $R_{xx}(t)$. Mean squared error will be used as a measure of quantity of the estimate. The problem then becomes,

$$\underset{\{c(i)\}}{\text{Minimize}}\ E[\,||R_{xx}(t) - \hat{R}_{xx}(t)||^2] \ . \tag{2.60}$$

For this preliminary investigation, values of $c(i)$ will be found which optimize the quality in (2.60), and then used to estimate the $P_{xd}(t)$ vector. A totally general approach would be to define some performance criterion which measures the accuracy of both the $R_{xx}(t)$ estimate and the $P_{xd}(t)$ estimate. This more general development will be pursued later this year. Since $R_{xx}(t)$ and $\hat{R}_{xx}(t)$ are

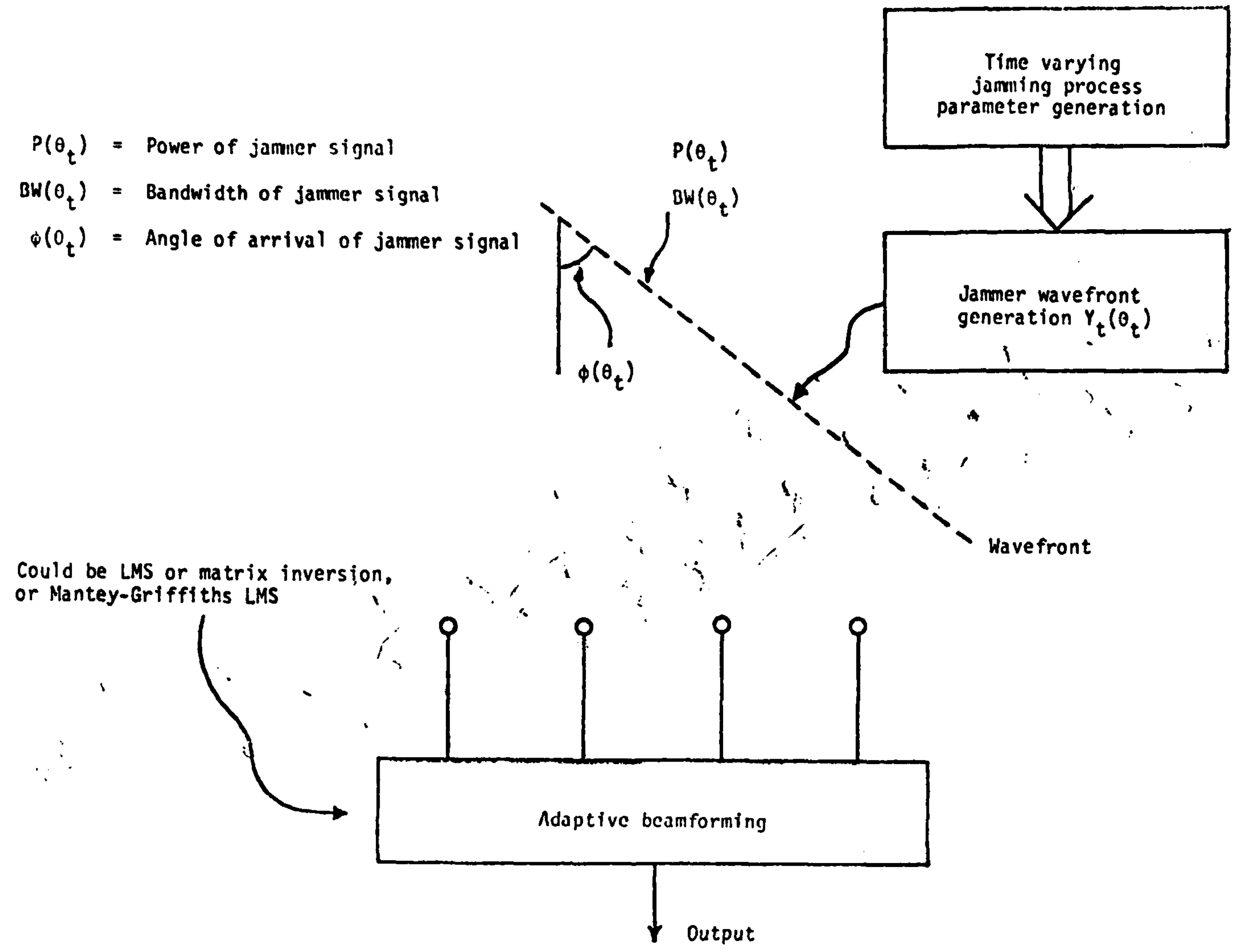

$P(\theta_t)$ = Power of jammer signal
$BW(\theta_t)$ = Bandwidth of jammer signal
$\phi(0_t)$ = Angle of arrival of jammer signal
Time varying jamming process parameter generation
Jammer wavefront generation $Y_t(\theta_t)$
$P(\theta_t)$
$BW(\theta_t)$
$\phi(\theta_t)$
Wavefront
Could be LMS or matrix inversion, or Mantey-Griffiths LMS
Adaptive beamforming
Output

Figure 2.12. Adaptive antenna and assumed jammer model used in estimation of time varying process covariance matrix.

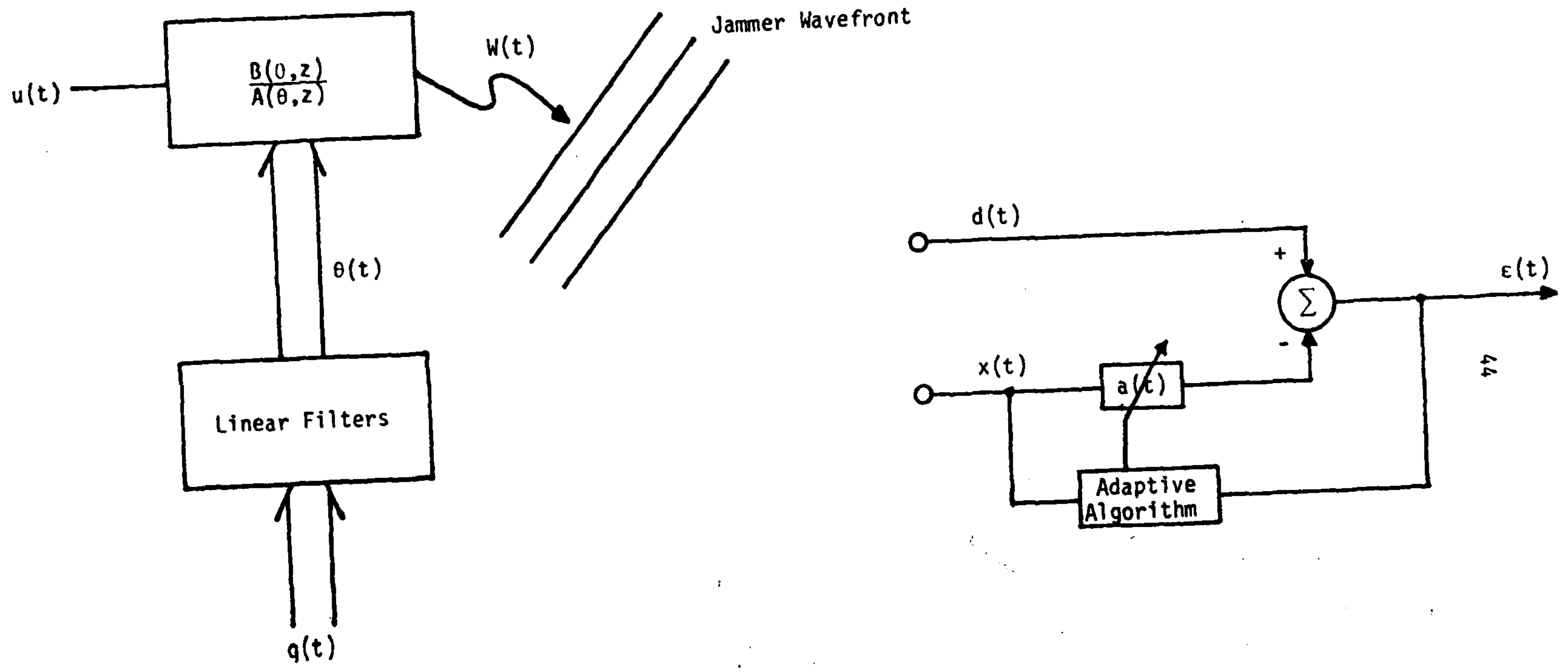


Figure 2.13. Specific jammer and antenna model.

matrices, a matrix norm must be chosen to give (2.60) meaning. A simple norm could be obtained by changing (2.60) to

$$\underset{\{c(i)\}}{\text{Minimize}}\ E[trace(R_{xx}(t) - \hat{R}_{xx}(t))^2] \quad . \tag{2.61}$$

A solution for (2.61) may be approximated by assuming the array to have only one auxiliary element, and solving

$$\underset{\{c(i)\}}{\text{Minimize}}\ E[\varphi_o(t) - \hat{\varphi}_o(t))^2] \tag{2.62}$$

where $\varphi_o(t)$ is the zeroth order time varying covariance lag,

$$\varphi_o(t) = E(x_1^2(t)|\vartheta_t) \tag{2.63}$$

and

$$\hat{\varphi}_o(t) = \sum_{i=t-l}^{t} c(t-i)X_1^2(i) \quad . \tag{2.64}$$

In making the above assumptions, the major simplification arises by performing the minimization on the squared error of the zeroth order lag. The assumption of only one auxiliary element does not decrease the generality of the result, but does decrease the notational complexity.

The solution to Eqs. (2.62) through (2.64) is a simple result of Wiener filter theory. Let

$$Z(t) = X_1^2(t) \quad , \tag{2.65}$$

and minimize (2.62) by setting its derivative with respect to the $C(i)$ equal to zero. This gives

$$\begin{aligned} 0 &= \frac{\partial}{\partial c(t-j)}(E(\varphi_o(t) - \hat{\varphi}_o(t))^2) \quad j = t-l, \dots, t \\ &= \frac{\partial}{\partial c(t-j)}(E(\varphi_o(t) - \sum_{i=t-l,}^{t} c(t-i)Z(i))^2) \quad j = t-l, \dots, t \\ &= 2E(Z(j)\varphi_o(t) - \sum_{i=t-l,}^{t} c(t-i)Z(i)Z(j)) \quad j = t-l, t \end{aligned} \tag{2.66}$$

or

$$\sum_{i=t-l}^{t} c(t-i)E\left\{Z(i)Z(j)\right\} = E\left\{\varphi_o(t)Z(j)\right\} \quad j = t-l, t \quad . \tag{2.67}$$

Equation (2.67) can be more efficiently stated in matrix notation as

$$R_{zz}C = P_{z\varphi} \tag{2.68}$$

or

$$C = R_{zz}^{-1} P_{z\varphi} \quad . \tag{2.69}$$

where

$$Z^T(t) = [z(t-l), z(t-l+1), \ldots, z(t)] \tag{2.70}$$

$$R_{zz} = E[Z(t)Z^T(t)] \tag{2.71}$$

$$P_{z\varphi} = E[Z(t)\varphi_o(t)] \tag{2.72}$$

$$C^T = [c(l-1), c(l-2), \ldots, c(1)] \quad . \tag{2.73}$$

Equation (2.68) is a large, Toeplitz set of equations. Typical values of l are about 200, which allows for a good approximation of the AR part of the estimator (see discussion following Eq. (2.52)). The quantities R_{zz} and $P_{z\varphi}$ are found from the statistics of the time varying parameters of the jammer.

Examples of solutions to (2.69) for certain assumed jammer models will now be given.

2.7 DETERMINING AN OPTIMAL WINDOW FOR A SIMPLE NON-STATIONARY JAMMER

An optimal window will now be found for a jammer using a single zero process to generate its wavefront. Figure 2.14 shows the antenna configuration, and how the jammer wavefront is generated. The jammer wavefront is given by

$$W(t') = u(t') + a_1(t')u(t'-1) \tag{2.74}$$

where $u(t')$ is a white noise sequence. Note that the same antenna configuration outlined in section 2.3 is used here. This configuration will be used throughout this part of the report to allow a common basis for comparison. The signal at the auxiliary element would then be,

$$x(t) = u(t'-n) + a_1(t'-n)u(t'-n-1) \; . \tag{2.75}$$

For simplicity, shift the time index relative to x, so that $t = t'-n$, giving

$$x(t) = u(t) + a_1(t)u(t-1) \; . \tag{2.76}$$

As outlined previously, $a_1(t)$ is generated by a time invariant stochastic process.

This formulation of the problem can now be solved using the techniques presented in section 2.6. First form R_{xx}, then find $P_{x\varphi}$, and solve the resulting set of linear equations. Expressions for the R_{xx} matrix and the $P_{x\varphi}$ vector will be found for each component. These components will then be reassembled to find the matrix and vector.

An expression for R_{xx} will be given as a function i,j, which indicate which row and column component (respectively) this expression represents. Both i and j range from $t-l$ to t since R_{xx} is l by l matrix. First,

$$\begin{aligned} R_{xx}(i,j) &= E\left\{x(i)x(j)\right\} \\ &= E\left\{x^2(i)x^2(j)\right\} \\ &= E\left\{(u(i) + a_1(i)u(i-1))^2(u(j) + a_1(j)u(j-1))^2\right\} \; . \end{aligned} \tag{2.77}$$

and assume $u(t)$ is a white noise sequence,

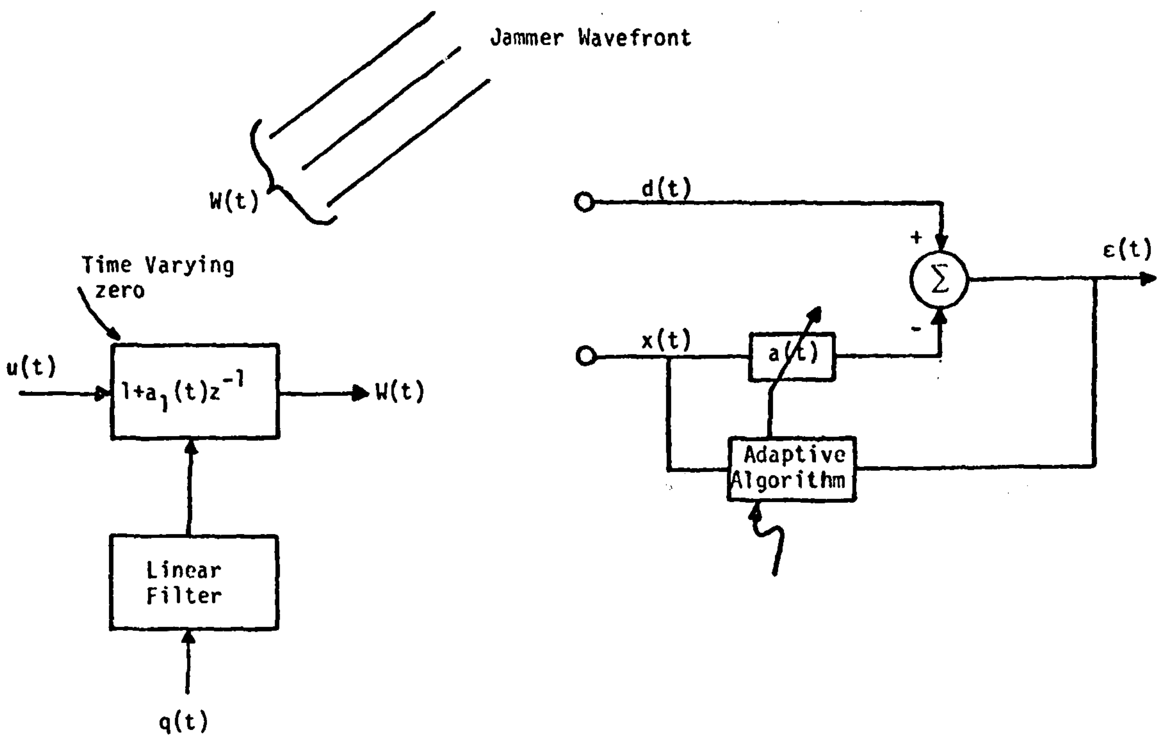


Figure 2.14. Single zero jammer wavefront generation.

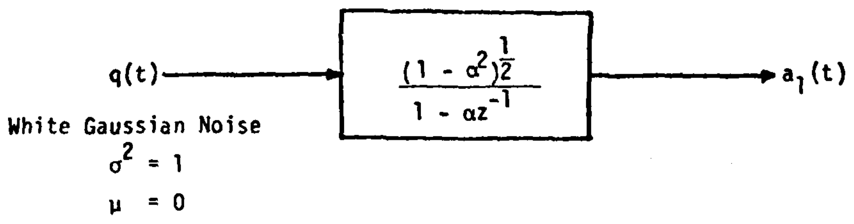


Figure 2.15. Model of first order Markov process generating $a_1(t)$.

$$E[u(i),u(j)] = \begin{cases} 0 & i \neq j \\ \overline{u^2} & i = j \end{cases} . \tag{2.78}$$

Multiplying out (2.77), distributing expectations, and substituting (2.78) gives

$$\begin{aligned} E[x^2(j)x^2(j)] = {} & E[u^2(i)u^2(j)] + E[a_1^2(i)u^2(i-1)u^2(j)] + \\ & 4E[a_1(i)a_1(j)u(i)u(j)u(i-1)u(j-1)] + \\ & E[a_1^2(j)u^2(i-1)u^2(j)] + E[a_1^2(i)a_1^2(j)u^2(i-1)u^2(j-1)] . \end{aligned} \tag{2.79}$$

Now assume $a(t)$ is uncorrelated with u and has zero mean. Denote the expectation operator as

$$E(x) = \bar{x} . \tag{2.80}$$

which gives

$$\begin{aligned} E[x^2(i)x^2(j)] = {} & \overline{u^2(i)u^2(j)} + \overline{a_1^2(i)u^2(i-1)u^2(j)} \\ & + 4\overline{a_1(i)a_1(j)u(i)u(j)u(i-1)u(j-1)} \\ & + \overline{a_1^2(j)u^2(i-1)u^2(j)} + \overline{a_1^2(i)a_1^2(j)u^2(i-1)u^2(j-1)} \end{aligned} \tag{2.81}$$

Further, since $a(t)$ is a stationary sequence the value of $R_{xx}(i,j)$ depends only on $|i-j|$. From (2.81),

$$R_{xx}(i,j) = \begin{cases} \overline{u^4}(1+\overline{a_1^4}) + 6\overline{u^2}^2\overline{a_1^2} & |i-j|=0 \\ \overline{u^4}\overline{a^2} + \overline{u^2}^2(1+\overline{a^2}+\overline{a_1^2(i)a_1^2(j)}) & |i-j|=1 \\ \overline{u^2}^2(1+2\overline{a^2}+\overline{a_1^2(i)a_1^2(j)}) & |i-j|\geq 2 \end{cases} \tag{2.82}$$

An expression for $P_{x\varphi}$ will be found in a similar way. Component i of the $P_{x\varphi}$ vector will be denoted $P_{x\varphi}(i)$. The components are found as,

$$P_{x\varphi}(i) = E(x^2(i)P_t) \tag{2.83}$$

where i ranges from t to $t-l$. P_t is the expected value of the power of x given knowledge of the value of the stochastic parameter $a_1(t)$.

$$P_t = E\left[x^2(t)|a_1(t)\right]$$

$$= E\left[(u(t) + a_1(t)u(t-1))^2 | a_1(t)\right]$$
$$= \overline{u^2}(t) + a_1^2(t)\overline{u^2}(t-1)$$
$$= \overline{u^2}\left[1 + a_1^2(t)\right] \qquad (2.84)$$

Now find $P_{x\varphi}(i)$ as

$$P_{x\varphi}(i) = E[x^2(i)\overline{u^2}(1 + a_1^2(t))]$$
$$= E[(u(i) + a_1(i)u(i-1))^2\overline{u^2}(1 + a_1^2(t))] \quad . \qquad (2.85)$$

Making the same assumptions used in obtaining (2.81), gives

$$P_{x\varphi}(i) = \overline{u^2}^2(1+2\overline{a_1^2}+\overline{a_1^2(i)a_1^2(t)}) \qquad (2.86)$$

Since it is desired to find the C vector which solves

$$R_{xx}C = P_{x\varphi} \, . \qquad (2.87)$$

The matrices R_{xx} and $P_{x\varphi}$ can be scaled by an arbitrary constant without affecting the solution. Dividing by $\overline{u^2}^2$ gives

$$R'_{xx}(i,j) = \begin{cases} K_u(1+\overline{a_1^4}) + 6\overline{a_1^2} & |i-j|=0 \\ K_u\overline{a^2} + 1 + \overline{a_1^2} + \overline{a_1^2(i)a_1^2(j)} & |i-j|=1 \\ 1 + 2\overline{a_1^2} + \overline{a_1^2(i)a_1^2(j)} & |i-j|>1 \end{cases} \qquad (2.88)$$

and

$$P'_{x\varphi}(i) = 1 + 2a_1^2 + \overline{a_1^2(i)a_1^2(t)} \qquad (2.89)$$

where K_u is the kurtosis of u, given by

$$K_u = \frac{\overline{u^4}}{\overline{u^2}^2} \, . \qquad (2.90)$$

Next, some model for how the $a_1(t)$ sequence is generated must be assumed. Often the $a_1(t)$ sequence will be a first-order Markov process, which can be approximated by driving a single pole filter with white gaussian noise,

$$a_1(t) = \alpha a_1(t-1) + (1-\alpha^2)^{1/2}q(t) \qquad (2.91)$$

where $q(t)$ is the white gaussian noise sequence. The value of $a_1(t)$ is a function of its last value and the random $q(t)$ sequence, hence the name first order Markov. Figure 2.15 illustrates the generation of $a_1(t)$ in the z-transform notation of equation (2.45). The autocovariance of $a_1(t)$ is given by

$$E[(a_1(i)a_1(j))] = \alpha^{|k|} \tag{2.92}$$

where $k = i-j$. A derivation of this result is presented in [14]. Note also that since $q(t)$ is a gaussian sequence, the output of the linear filter $a(t)$ will also be gaussian (this result can be found in [13]). Because $a_1(t)$ is gaussian, its fourth order moments can be broken down as,

$$E[(a^2(i)a^2(j))] = E[(a^2(i))E(a^2(j))] + 2\Big[E[(a_1(i)a_1(j))]\Big]^2 \quad . \tag{2.93}$$

Substituting (2.92) gives

$$E[a^2(i)a^2(j)] = 1 + 2\alpha^{2|k|} \quad . \tag{2.94}$$

Using this result, (2.88) becomes

$$R'_{xx}(i,j) = \begin{cases} k_u(1+k_a) + 6 & |i-j| = 0 \\ k_u + 3 + 2\alpha^{2|i-j|} & |i-j| = 1 \\ 4 + 2\alpha^{2|i-j|} & |i-j| > 1 \end{cases} \tag{2.95}$$

and (2.89) becomes

$$P'_{x\varphi}(i) = 4 + 2\alpha^{2|i-t|} \tag{2.96}$$

Since $a_1(t)$ is gaussian $k_a = 3$.

There are two major points to be made about (2.95) and (2.96). First, the assumed first order Markov variation of $a_1(t)$ results in an exponential windowing form of the $c(i)$ when k_u is small. This result is found by solving

$$C = R'^{-1}_{xx}P'_{x\varphi} \tag{2.97}$$

using (2.95) and (2.96). The $c(i)$ turn out to be exponential with decay factor α^2. The $c(i)$ to not exactly follow decay for values of i near 0 or l, but these effects are due to the finite length window and are negligible. The point is that exponential windowing of data is optimal when the time varying parameters are first order Markov. The second point is to interpret the meaning of the kurtosis of u, and observe the effect k_u was on the optimal window. Since $u(t)$ is zero mean, k_u (see Eq. 2.96) can be viewed as a measure of the variance of the power of $u(t)$. A large k_u means that the power of $u(t)$ undergoes large changes in comaparison to its average power. A small value of k_u means the $u(t)$ process $u(t)$ was small deviations about its average power . Note the effect this has on the window, large k_u values force R_{zz} close to a tridiagonal form which means the $c(i)$ coefficients decrease slowly. This is intuitively understandable, since more data will be required to form a meaningful average of $u(t)$. Small k_u does just the opposite making the $c(i)$ coefficients decrease exponentially.

2.8 AN OPTIMAL WINDOW FOR A NON-SIMPLE NON-STATIONARY JAMMER

This section presents an optimal window for a jammer using a multiple zero filter to generate its wavefront (see Fig. 2.16). The angle of arrival (φ) is held constant, and the frequency components of the jammer are varied. The wavefront of the jammer is generated

$$W(t') = a_0(t')u(t') + \cdots + a_k(t')u(t'-k) \tag{2.98}$$

where $u(t')$ is a gaussian, white noise sequence. The signal at the auxilliary element is,

$$x(t) = a_0(t'-m)u(t'-m) + \cdots + a_k(t'-m)u(t'-k-m) \tag{2.99}$$

and redefining the time index as in Section 2.7 so that $t = t' - m$ gives

$$\begin{aligned} x(t) &= a_0(t)u(t) + \cdots + a_k(t)u(t-k) \\ &= \sum_{n=0}^{k} a_n(t)u(t-n) \ . \end{aligned} \tag{2.100}$$

The coefficients $a_i(t)$ are time invariant stochastic processes which represent the time varying nature of the jammer. As described before (section 2.5), this filter is a moving average (all zero) type, which means that for a fixed set of $a_i(t)$, $x(t)$ has a frequency spectrum with n dips in it. The specific character of the dips is a function of the $a_i(t)$, hence if the $a_i(t)$ are allowed to change, the spectral character of the jammer becomes the varying.

An optimal window for the data of the above described jammer can be found by following a similar collection of steps to those of section 2.7. First, form the R_{xx} and $P_{x\varphi}$ quantities, then solve for the optimal window. The R_{xx} matrix and $P_{x\varphi}$ vector will be determined in a component-wise fashion to make the derivation more clear. Once these component expressions are determined, they will be combined to form R_{xx} and $P_{x\varphi}$.

The R_{xx} matrix will now be found. Let the components of R_{xx} be denoted

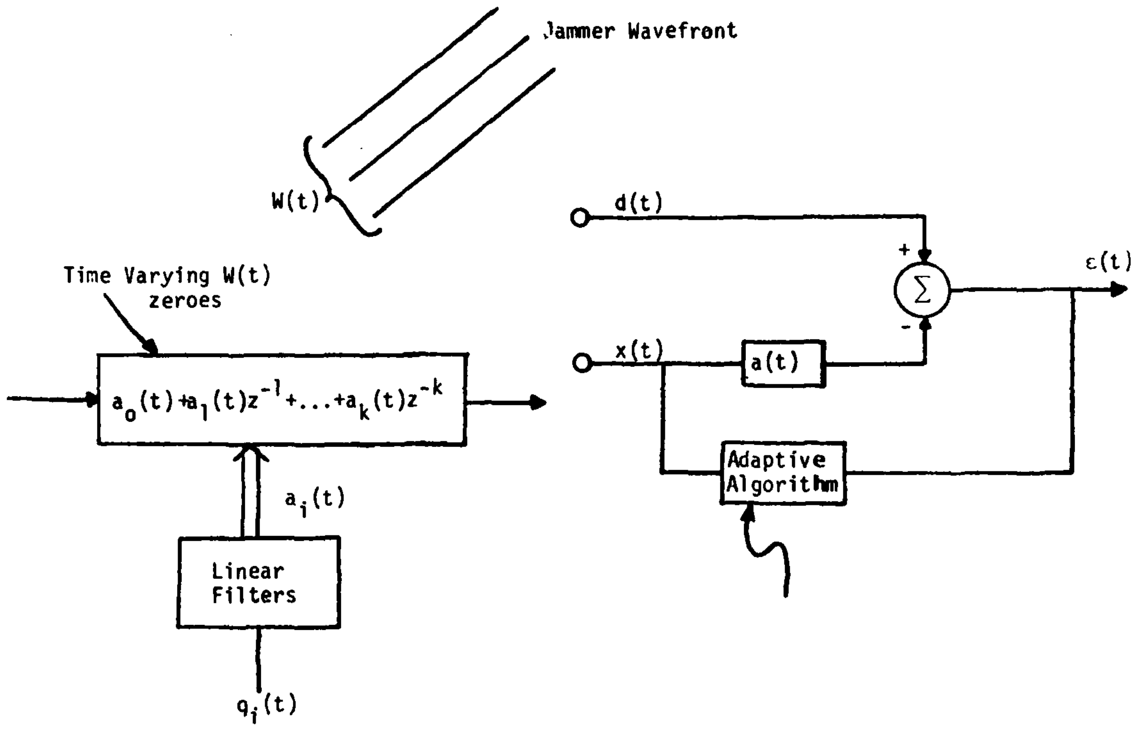
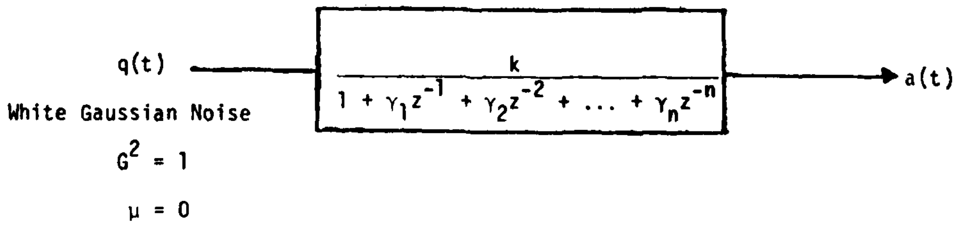


Figure 2.16. Multiple zero jammer wavefront generation.



q(t)
White Gaussian Noise
$G^2 = 1$
$\mu = 0$
$\frac{k}{1+\gamma_1 z^{-1}+\gamma_2 z^{-2}+\ldots+\gamma_n z^{-n}}$
a(t)

Figure 2.17. Model of n^{th} order Markov process generatinn a(t).

$R_{zz}(i,j)$, where i is the row index and j is the column index which locate the component in R_{zz}. Both i and j range from $t-l$ to t and R_{zz} is an l by l matrix

$$\begin{aligned} R_{zz}(i,j) &= E\left[z(i)z(j)\right] \\ &= E\left[E(z(i)z(j)\,|\,a_m(i)a_n(j))\right] \\ &= E\left[E(x^2(i)x^2(j)\,|\,a_m(i)a_n(j))\right] \end{aligned} \qquad (2.101)$$

The last equality of (2.101) indicates how R_{zz} will be found. R_{zz} is formed by first taking an expectation over x given the random $a_m(i)$ coefficients. Then an expectation with respect to the $a_m(i)$ coefficients is taken giving R_{zz}. The assumption that $u(t)$ is a gaussian sequence in (2.98) implies that $x(t)$ is also gaussian since it is a linear combination of past $u(t)$ values. This gives the simplification,

$$\begin{aligned} E[x^2(i)x^2(i)\,|\,a_m(i)a_n(j)] &= E[x^2(i)\,|\,a_m(i))E(x^2(j)\,|\,a_n(j)] + \\ & 2\left[E[x(i)x(j)\,|\,a_m(i)a_n(j)]\right]^2 . \end{aligned} \qquad (2.102)$$

Now,

$$x(i) = \sum_{m=0}^{k} a_m(i)u(i-m) \qquad (2.103)$$

and

$$x^2(i) = \sum_{m=0}^{k} \sum_{n=0}^{k} a_m(i)u(i-m)a_n(i)u(i-n) \qquad (2.104)$$

which results in

$$E[x^2(i)\,|\,a_m(i)] = \sum_{n=0}^{k} \sum_{n=0}^{k} a_m(i)a_n(i)E\left[u(i-m)u(i-n)\right] . \qquad (2.105)$$

Note that $E[u(i-m)u(i-n)]$ is nonzero only when $i-m = i-n$. So (2.105) simplifies to

$$E[x^2(i)\,|\,a_m(i)] = \sum_{n=0}^{k} a_m^2(i)\overline{u^2} . \qquad (2.106)$$

Evaluating the second term in (2.102) gives,

$$E\left[x(i)x(j)\,|\,a_m(i)a_n(j)\right] = \sum_{m=0}^{k}\sum_{n=0}^{k} a_m(i)a_n(j)E\left[u(i-m)u(j-n)\right] \qquad (2.107)$$

where $E\left[u(i-m)u(j-n)\right]$ is nonzero only when $i-m = j-n$. This lets (2.107) simplify to

$$E[x(i)x(j)\,|\,a_n(i)a_n(j)] = \sum_{m=0}^{k-l'} a_m(i)a_{m+l}(j)\overline{u^2} \qquad (2.108)$$

where $l' = i-j$ and $l' \le k$. For $i > k$, $E\left[x(i)x(j)\,|\,a_m(i)a_n(j)\right] = 0$.

Now evaluate $R_{zz}(i,j)$ by taking expectations of (2.106), and (2.108). This will be done in two parts. R_{zz} is l by l so $l' = i-j$ ranges from 0 to l . The length, k, of the all zero jammer wavefront generation process is not related to the length of the window, l. Hence k can be greater or smaller than l. The following development covers values of $R_{zz}(i,j)$ for any k,l. The two cases to consider are $l' \le k$ and $i > k$. First, for $l' \le k$,

$$E[z(i)z(j)] = E\left[\sum_{m=0}^{k}\sum_{n=0}^{k} a_m^2(i)a_n^2(j)\overline{u^{2^2}}\right] +$$
$$+E\left[2\overline{u^{2^2}}\sum_{m=0}^{k-l'}\sum_{n=0}^{k-l'} a_m(i)a_{m+l'}(j)a_n(i)a_{n+l'}(j)\right] \qquad (2.109)$$

Denoting expectation by over-bars as in Eq. (2.80) of gives

$$E[z(i)z(j)] = \overline{u^{2^2}}\left[\sum_{m=0}^{k}\sum_{n=0}^{k}\overline{a_m^2(i)a_n^2(j)} + \right.$$
$$\left. + 2\sum_{m=0}^{k-l'}\sum_{n=0}^{k-l'}\overline{a_m(i)a_{m+l'}(j)a_n(i)a_{n+l'}(j)}\right] \qquad (2.110)$$

Now, for $l' > k$,

$$E[z(i)z(i)] = \overline{u^{2^2}}\sum_{m=0}^{k}\sum_{n=0}^{k}\overline{a_m^2(i)a_n^2(j)} \qquad (2.111)$$

To achieve a further simplification, assume that the sequences $a_m(i)$ and $a_n(j)$ are uncorrelated for $m \ne n$ and that they are zero mean, which makes

$$\overline{a_m(i)a_n(j)} = 0 \text{ for } m \neq n \quad . \tag{2.112}$$

Substituting this into (110) and (111) gives,

$$E[z(i)z(j)] = \begin{cases} \overline{u^2}^2 \left[\sum_{m=0}^{k} \sum_{n=0}^{k} \overline{a_m^2(i)a_n^2(j)} + 2 \sum_{m=0}^{k} \sum_{n=0}^{k} \overline{a_m^2(i)a_n^2(j)} \right] & l' = 0 \\ \overline{u^2}^2 \left[\sum_{m=0}^{k-l'} \sum_{n=0}^{k-l'} \overline{a_m^2(i)a_n^2(j)} + 2 \sum_{m=0}^{k-|l'|} \overline{a_m^2(i)a_{m+|l'|}^2(j)} \right] & 0 < |l'| < k \\ \overline{u^2}^2 \left[\sum_{m=0}^{k} \sum_{n=0}^{k} \overline{a_m^2(i)a_n^2(j)} \right] & k \leq |l'| \end{cases} \tag{2.113}$$

Next assume that the $a_m(i)$ are all identically distributed random variables, which gives the final simplification. For $i-j = 0$,

$$E[z(i)z(j)] = \overline{u^2}^2 \left[3 \sum_{m=0}^{k} (\overline{a^4(t)} - \overline{a^2(t)}^2) + 3 \sum_{m=0}^{k} \sum_{n=0}^{k} (\overline{a^2})^2 \right]$$

$$= \overline{u^2}^2 \; 3(k+1)(\overline{a^4} + k(\overline{a^2})^2) \quad . \tag{2.114}$$

For $1 \leq |i-j| = |l'| \leq k$

$$E[z(i)z(j)] = (\overline{u^2})^2 \left[\sum_{m=0}^{k} (\overline{a^2(i)a^2(j)} - \overline{a^2}^2) + \sum_{m=0}^{k} \sum_{n=0}^{k} \overline{a^2}^2 + 2 \sum_{m=1}^{k-|l'|} (\overline{a^2}^2) \right]$$

$$= \overline{u^2}^2 \left[(k+1)\, \overline{a^2(i)a^2(j)} + k(k+1)\overline{a^2}^2 + 2(k-|l'|+1)\overline{a^2}^2 \right] \tag{2.115}$$

and for $k < |l'|$

$$E(z(i)z(j)) = \overline{u^2}^2 \left[\sum_{m=0}^{k} (\overline{a^2(i)a^2(j)} - \overline{a^2}^2) + \sum_{m=0}^{k} \sum_{m=0}^{k} \overline{a^2}^2 \right]$$

$$= \overline{u^2}^2 \left[(k+1)\overline{a^2(i)a^2(j)} - (k+1)\overline{a^2}^2 + (k+1)^2(\overline{a^2})^2 \right] \quad . \tag{2.116}$$

Since $a(i)$ is a stationary sequence, let

$$\varphi_{bb}(l') = \overline{a^2(i)a^2(j)} \ , \quad l' = i-j \ . \tag{2.117}$$

Combining 2.114, 2.115, 2.116 and 2.117 gives,

$$(\overline{u^2})^2 E[z(i)z(j)] = \begin{cases} 3(k+1)(\overline{a^4} + k\overline{a^2}^2) & l' = 0 \\ (k+1)\varphi_{bb} l' + (k^2+3k-2(l')+2)(\overline{a^2})^2 & 1 \le l' \le k \\ (k+1)\varphi_{bb}(l') + (k^2+k)\overline{a^2}^2 & k < l' \end{cases} \tag{2.118}$$

Next the elements of $P_{x\varphi}$, denoted as $P_{x\varphi}(i)$, will be found. The index i ranges from $t-l$ to t and indicates where the component belongs in the $P_{x\varphi}$ vector. First,

$$\begin{aligned} P_{x\varphi}(i) &= E(x^2(i)P_t) \\ &= E[E(x^2(i)P_t \,|\, a_m(i))] \ , \end{aligned} \tag{2.119}$$

where

$$\begin{aligned} P_t &= E[x^2(t) \,|\, a_m(i)] \\ &= \sum_{m=0}^{k} a_m^2(t)\overline{u^2} \ . \end{aligned} \tag{2.120}$$

The last equality is the result of Eq. (2.106). To evaluate (2.119), an expression for the conditional expectation term must be found. Using Eq. (2.120) gives

$$E[x^2(i)P_t \,|\, a_m(i)] = \sum_{m=0}^{k} \sum_{n=0}^{k} a_m^2(i)a_n^2(t)\overline{u^2}^2 \ , \tag{2.121}$$

since the $u(t)$ sequence is white. Now taking expectation with respect to $a_m(i)$ yields the total expectation,

$$E[x^2(i)P_t] = \overline{u^2}^2 \sum_{m=0}^{k} \sum_{n=0}^{k} \overline{a_m^2(i)a_n^2(t)} \ . \tag{2.122}$$

Now, as before, assume that the $a_m(i)$ are uncorrelated, zero mean random variables, which results in the condition shown in equation 2.112. Also assume that the $a_m(i)$ are identically distributed for all m. This leads to a simplification of (2.122) to

$$E[x^2(i)P_t] = \begin{cases} \overline{u^2}^2((k+1)\overline{a^4} + k(k+1)\overline{a^2}^2) & i = k \\ \overline{u^2}^2((k+1)\varphi_{bb}(t-i) + k(k+1)\overline{a^2}^2) & i < k \end{cases} \tag{2.123}$$

Equations (2.118) and (2.123) represent component descriptions of R_{xx} and $P_{x\varphi}$ in terms of the values of $\overline{a^2}, \overline{a^4}$ and $\varphi_{bb}(i)$. These quantities are all determined by the dynamics of the $a_i(t)$ coefficients. Since it was assumed that the $a_i(t)$ are independent, and identically distributed, only one model is required to find the unknown statistics of the $a_i(t)$. The random variable $a(t)$ is used to represent the statistics of all the $a_i(t)$. A good model of how these coefficients change is to assume the $a(t)$ is an n-th order Markov process, that is,

$$a(t) = \gamma_1 a(t-1) + \cdots + \gamma_n a(t-n) + q(t) \ . \tag{2.124}$$

where $q(t)$ is a gaussian white noise with zero mean. Figure 2.17 shows the z-transform representation of how $a(t)$ is generated. The actual $a_i(t)$ sequences used for simulation are generated as shown in Fig. 2.17, but to make them independent, k independent $q(t)$ sequences are presented to k separate ,but identical, filters (see Fig. 2.18). Since the gain k is variable, the $q_i(t)$ sequences can be restricted to being unit variance with no loss of flexibility. Since $q(t)$ is gaussian, $a(t)$ is gaussian since the filter acts as a linear operator. To find $\varphi_{bb}(i-j)$, the fact that $a(t)$ is gaussian gives

$$\begin{aligned} \varphi_{bb}(i-j) &= E(a^2(i)a^2(j)) \\ &= E(a^2(i))E(a^2(j)) + 2(E(a(i)a(j)))^2 \\ &= (\overline{a^2})^2 + 2\varphi_{aa}^2(i-j) \ . \end{aligned} \tag{2.125}$$

The $a(t)$ sequence is stationary and therefore $\varphi_{aa}(i-j)$ represents the auto covariance matrix of $a(t)$. Values of $\varphi_{aa}(i-j)$ may be found by the spectral factorization theorem [14] as

$$\varphi_{aa}(k) = \text{Inverse Z Transform} \left\{ \frac{k^2}{(1 - z^{-1}\gamma(z))(1 - z\gamma(z^{-1}))} \right\} \ . \tag{2.126}$$

The point is that $\varphi_{aa}(k)$ is readily obtainable, given the $\gamma(z)$ polynomial.

Since $a(t)$ is gaussian, it is known that $\overline{a^4} = 3\overline{a^2}^2$. These results may be substituted into equations 2.118 and 2.123 to get,

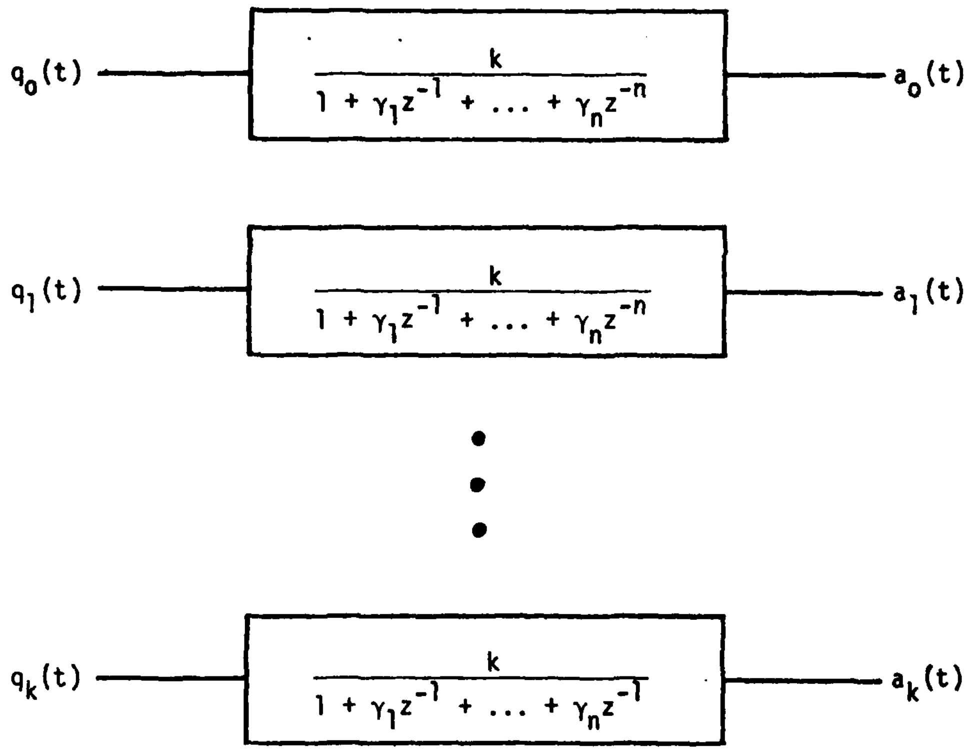

$q_o(t)$
$\frac{k}{1 + \gamma_1 z^{-1} + \ldots + \gamma_n z^{-n}}$
$a_o(t)$
$q_1(t)$
$\frac{k}{1 + \gamma_1 z^{-1} + \ldots + \gamma_n z^{-n}}$
$a_1(t)$
$q_k(t)$
$\frac{k}{1 + \gamma_1 z^{-1} + \ldots + \gamma_n z^{-1}}$
$a_k(t)$

Figure 2.18. Expanded view of 'linear filters' box of Fig. 2.18.

$$\overline{u^2}^{-2}\; E(x^2(i)x^2(j)) \;=\; \begin{cases} 3(k+a)(3+k)\overline{a^2}^2 & 0 = l' \\ 2(k+1)\varphi_{aa}^2(l') + (k^2+4k-2|l'|+3)\overline{a^2}^2 & 1 \le l' \le k \\ 2(k+1)\varphi_{aa}^2(l') + (k^2+2k+1)\overline{a^2}^2 & k < l' \end{cases} \qquad (2.127)$$

and

$$\overline{u^2}^{-2}\; E(x^2(i)P_t) \;=\; \begin{cases} (k+1)(3+k)\overline{a^2}^2 & i = t \\ (k+1)\varphi_{aa}^2(t-i) + (k+1)^2\overline{a^2}^2 & i < t \end{cases} \qquad (2.128)$$

Using the above equations, an optimal window was found for the process

$$a_i(t) \;=\; 1.98a_i(t-1) + .9801\; a_i(t-2) \;+ q_i(t) \qquad (2.129)$$

which is shown in Fig. 2.19. The same setup as described in section 2.3 is used for a performance comparison between an algorithm using an optimal window and an algorithm using an exponential window. Figure 2.20 displays the result. The antenna array employing an optimal windowing strategy tracks the changes in the true parameter more closely. The value of the decay constant, α, for the exponential window was determined so that both algorithms had the same weight variance. Further, Fig. 2.21 shows another comparison where the exponential windowing algorithm was allowed to have a higher weight variance than in Fig. 2.20. Note that even in this case, the tracking properties of the optimal windowing algorithm are superior.

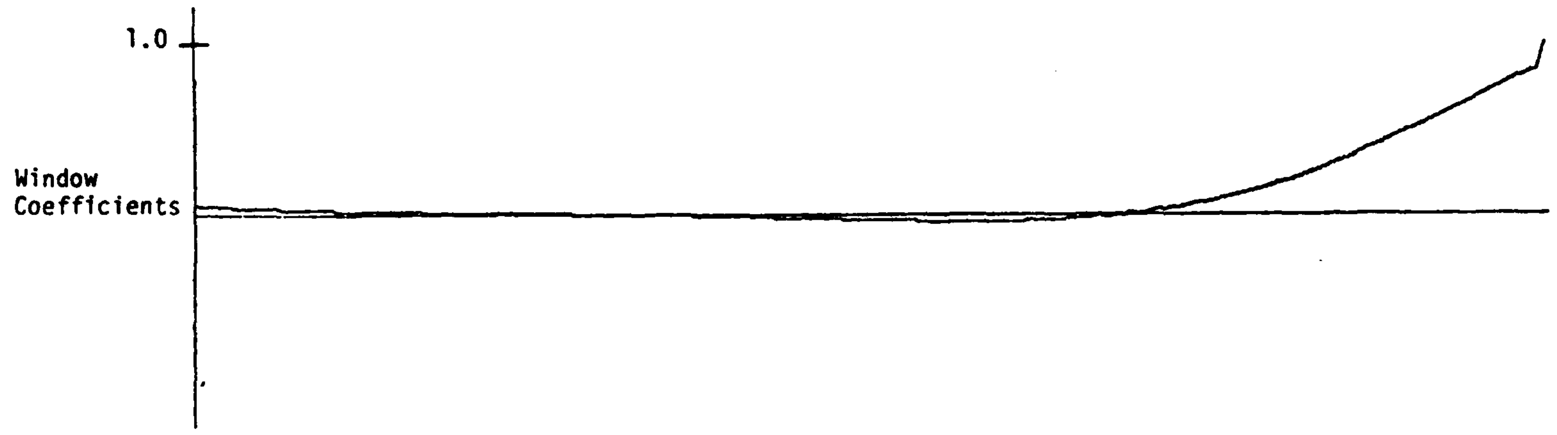


Figure 2.19. Optimal window for time varying jammer of section 2.8.

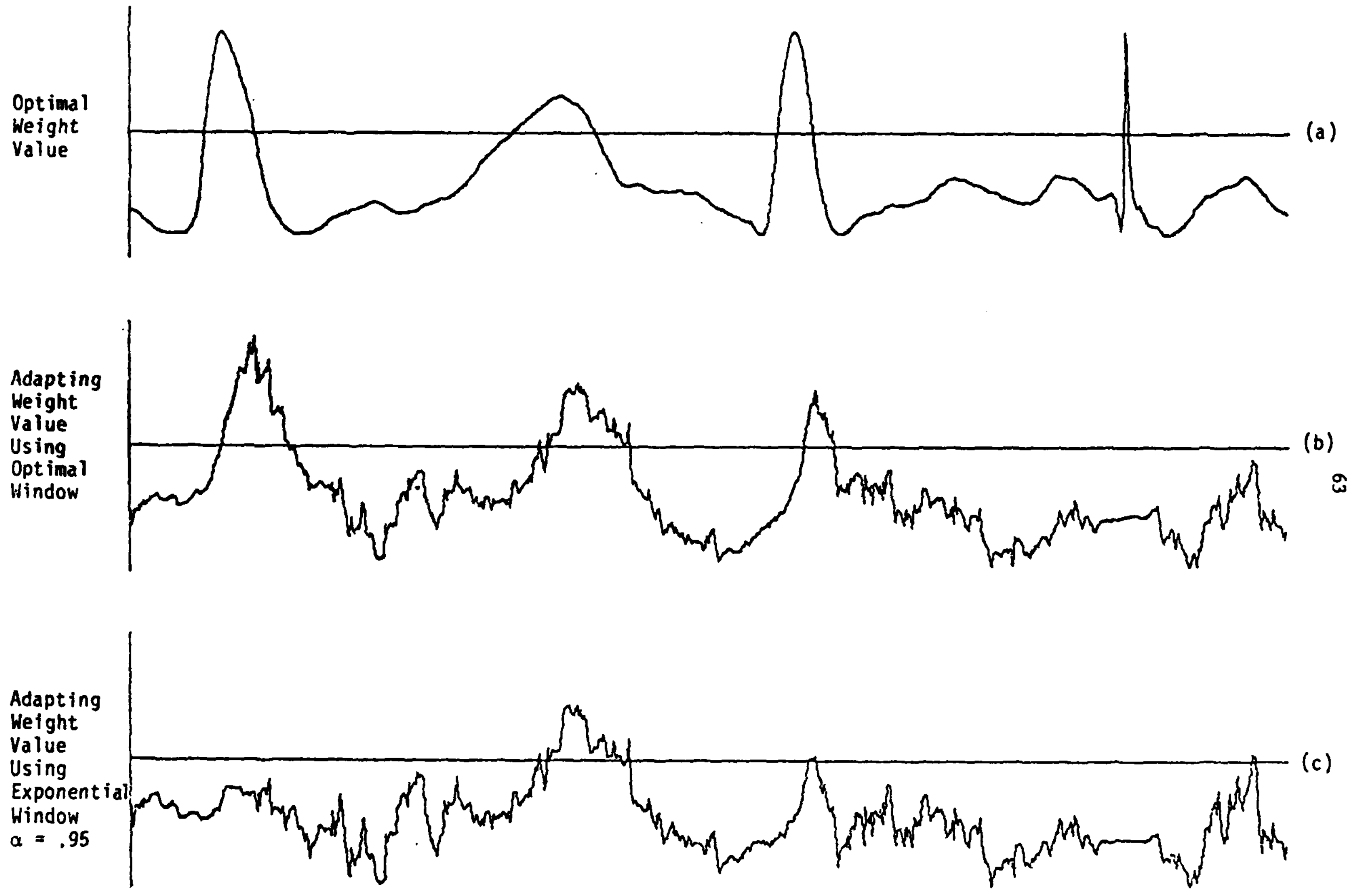


Figure 2.20. Comparison of optimal and exponential window algorithms.

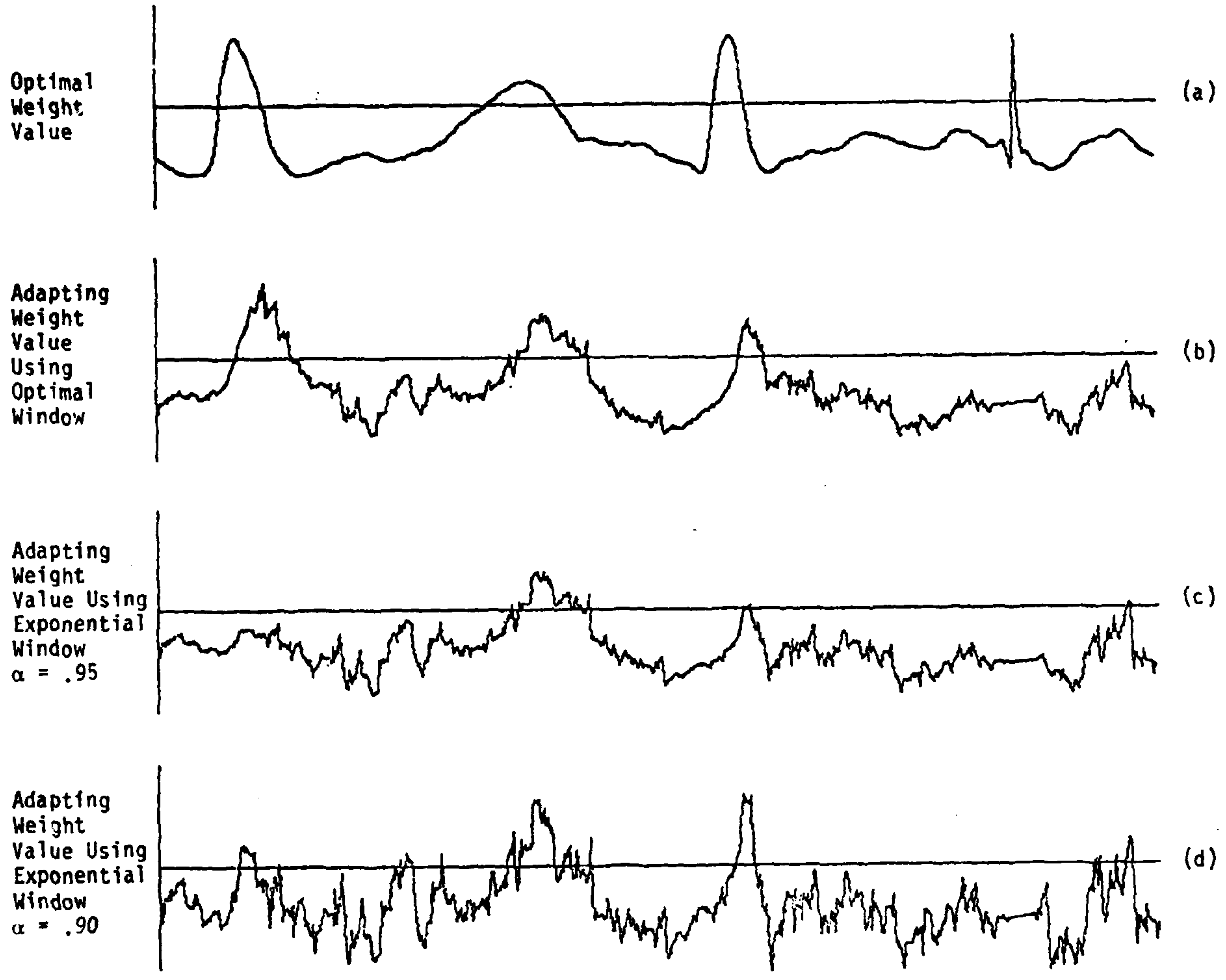


Figure 2.21. Illustration of improvement in misadjustment and tracking tradeoff by use of optimal window.

2.9 CONCLUSION AND SUGGESTIONS FOR FURTHER STUDY

The weighted recursive least squares algorithm has been derived in the context of realizing a Zahm adaptive beamformer. This algorithm has been shown to be a practical implementation of the ideal LMS/Newton algorithm. The ability of the algorithm to track a time-varying jammer was shown to be highly dependent on the exponential decay rate of the weighting coefficients. Minimization of mean squared error averaged over the nonstationarity was found to involve a tradeoff between weight variance and weight tracking speed. The restriction of considering only exponential weighting of the outputs was then removed and a method for finding a more complex, higher order optimal weighting was given in sections 2.5 and 2.6. Section 2.7 demonstrated that the use of an exponential weighting is optimal when the parameters of the jammer are changing as a first order Markov process. Secion 2.8 demonstrated that the use of an optimal weighting on the error outputs results in an improvement of the tradeoff between parameter variance and parameter tracking. Combining these results, if the jammer is believed to have a simple time varying nature, exponential weighting may be employed by a recursive algorithm with near optimal result. If the jammer is thought to have a complicated time varying structure, it would be worthwhile to find an optimal data weighting and use this for implementing the adaptive algorithm. Considerable improvements in performance could result.

Further study will involve defining a more general estimation error criterion than the zeroth-order lags proposed here. Also, this performance criterion should include some contribution from errors in estimation of the cross correlation vector P_{xy} (of section 2.4). Work should also be done in developing a recursive algorithm such as the iterative inverse using the higher order weighting techniques. Finally these results should be extended to an n-element Zahm array and other types of adaptive arrays.

PART III

IIR BEAMFORMING

3.1 Introduction

Another way to improve the effectiveness and lower the cost of an adaptive beamformer would involve the use of a new underlying beamformer structure. In most broadband beamformers the basic structure is that of a multi-channel FIR (zeros only) digital filter. In this section we describe a beamformer that is based on both IIR and FIR (both poles and zeros) filtering. It will be shown that an IIR beamformer gives improved nulling performance over the conventional FIR beamformer. As the hardware becomes available, beamformers based on IIR filtering may prove to be extremely valuable to the beamforming community.

3.2 Broadband Beamforming

A narrowband beamformer uses a complex gain at each element to perform the antenna weighting. A complex gain is realized by using a 90 degree phase shifter to split the signal path into seperate in-phase and quadrature phase channels. Each channel is then fed through selectable attenuators corresponding to the real and imaginary part of the complex gain. Finally, the two channels are recombined by summation. A beamformer using complex weighting can null interference which is of narrowband nature only and is therefore useful only in systems where the desired signal is narrowband.

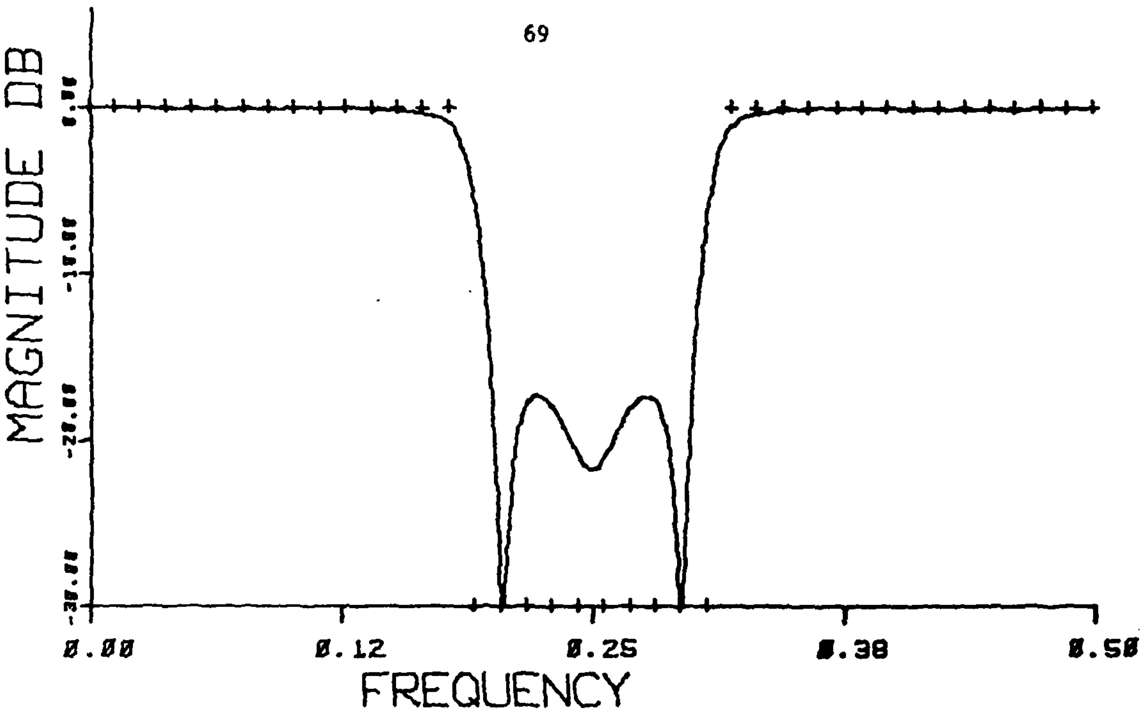
In a broadband communication system such as spread-spectrum, it is necessary to use antennas operating over a much broader bandwidth. Broadband adaptive antennas were first proposed in [2]. These systems differ from the narrowband beamformer by replacing the complex weights with tap delay line filters. A tap delay line filter is commonly refered to as a finite impulse response or FIR filter. A beamformer using FIR filters rather than complex weights is able to use temporal as well as spatial information to eliminate unwanted signals and to pass the signal of interest.

Another class of digital filters are those possessing an infinite impulse response and are appropriately called IIR filters. An IIR filter in many cases will require far fewer weights than an FIR filter to achieve an equally effective frequency response. In figure 3.1, the frequency response of an FIR and an IIR digital filter are contrasted where the desired chacteristic is to pass all frequencies except those in a narrow region. For equal number of weights, notice the improved performance of the IIR filter over the FIR filter. With these results in mind, we set out to investigate the application of IIR filters to adaptive beamforming. When adapting the weights of an IIR filter to achieve a desired effect, one encounters many problems. In the signal processing field, there have been many attempts over the past ten years or so to try to develop a reliable algorithm for adaptive IIR filtering [15,16]. Recently, we have developed a novel technique for implementing an IIR adaptive filter and are investigating it in applications to digital filter design, adaptive contol, adaptive noise canceling, adaptive line enhancement and to adaptive beamforming.

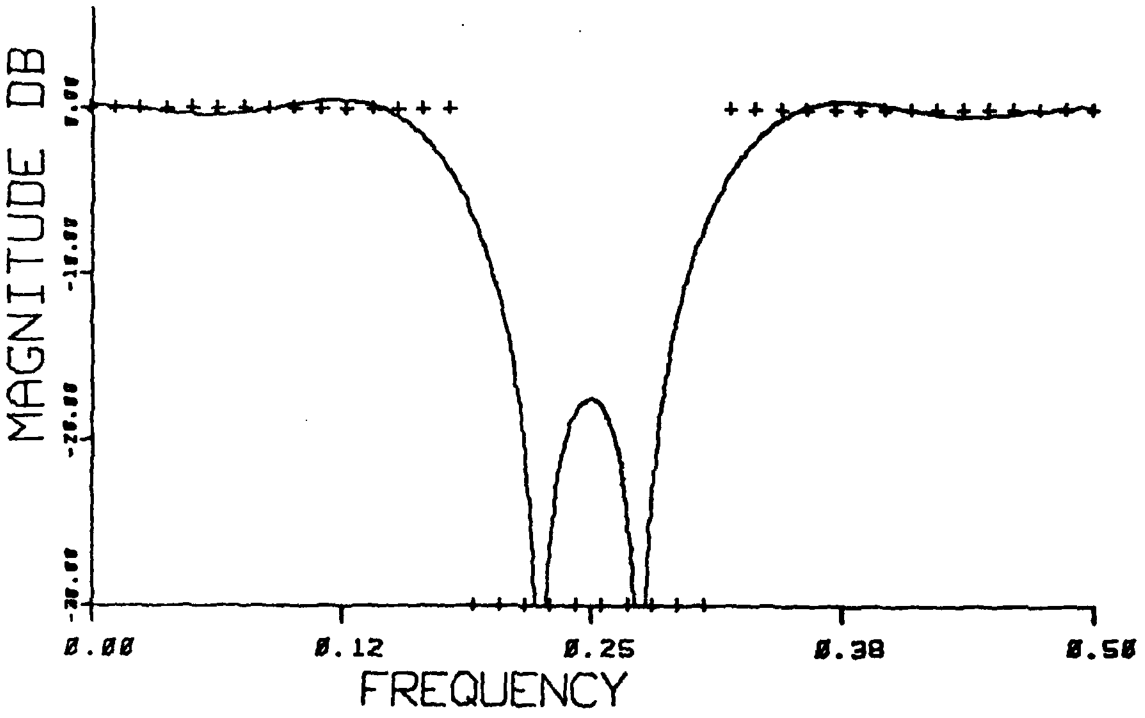
In the next section, we will briefly describe the Zahm beamformer and show its similarity to a general purpose adaptive filter. We will then describe our new technique of IIR adaptive filtering and give simulation results demonstrating the performance of an IIR Zahm beamformer.

3.3 Broadband Zahm Beamformer

A two element Zahm beamformer is shown in figure 3.2. The signal to be received is assumed to be of low power compared to interference (jamming) signals. The beamformer acts like a signal power inverter causing high powered jamming signals to be attenuated relative to the low power desired signal. It operates by minimizing the antenna output power subject to a "soft constraint". The primary antenna element is coupled directly, and it maintains the soft omni-directionality constraint and keeps the beamformer output power from



(a)



(b)

Figure 3.1. Comparison of (a) IIR and (b) FIR band reject filters having equal numbers of degrees of freedom.

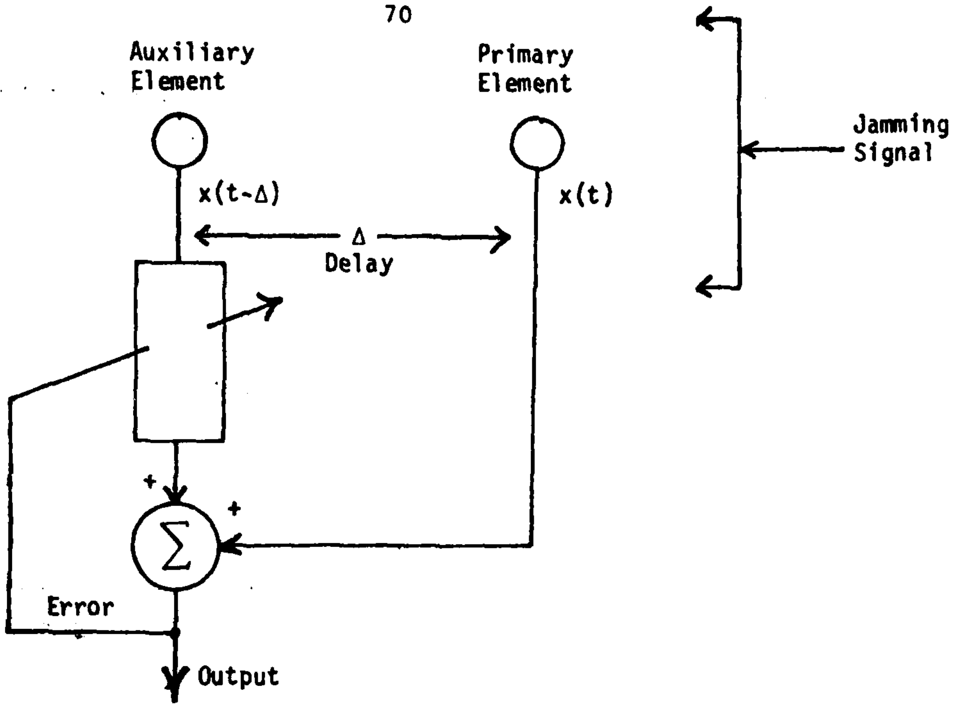


Figure 3.2a. Simple form of a broadband Zahm beamformer.

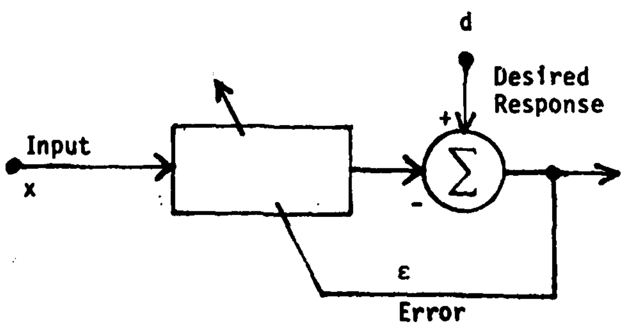


Figure 3.2b. General purpose adaptive filter.

being driven to zero. Attached to the auxilary elements are FIR adaptive filters. These filters allow the flexibility of frequency filtering as well as spatial filtering to remove unwanted jamming signals. Figure 3b shows a general purpose adaptive filter. The two-element Zahm beamformer is simply an adaptive filter whose desired response and primary input are connected to the primary antenna element and the auxilliary antenna element respectively.

3.4 IIR Adaptive Filtering

Keeping in mind the simple connection between adaptive beamforming and adaptive filtering, we now discuss a general purpose adaptive IIR filter. The usual form of FIR adaptive filter uses LMS algorithm to adjust its weights to minimize the mean square error between the desired response and the filter output. The FIR LMS adaptive filter has been used in a wide range of applications with favorable results. There has been much research concerning the development of a general purpose adaptive IIR filter possessing the same robust characteristic as the FIR LMS algorithm.

In this section we will describe our technique for implementing an IIR adaptive filter. Figure 3.3 shows the basic structure of an IIR adaptive filter. We use z-transform notation to indicate a linear filter, the converged adaptive filter. The polynomial B(z) corresponds to the zeros or FIR part of the digital filter and the polynomial $\frac{1}{1 + z^{-1}A(z)}$ corresponds to the poles or IIR part of the filter. The fixed term 1 in the denominator polynomial alleviates the problem of dividing by zero and does not influence the generality of the total structure. The adaptive part of this filter adjusts the coefficients of the A(z) and B(z) polynomial in such a manner that the mean squared output error is minimized. This is standardly refered to as an " output error " method since it is directly minimizing the difference between the desired response d and the filter output y.

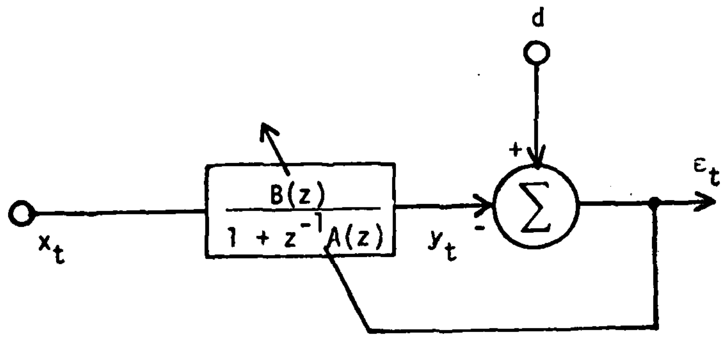


Figure 3.3. Basic IIR adaptive filter.

The output error method of adaptive IIR filtering has many problems. The mean square error is a non-quadratic function of the feedback weights. This causes major difficulties that lie in maintaining algorithm stability, providing relatively rapid convergence rates and guaranteeing optimality of the converged filter (i.e. convergence to the global rather than to a local optimum). In the FIR case, these problems are for the most part, nonexistent.

For example, a fixed FIR filter is always stable. Since the filter output is a linear sum of delayed versions of its input, a bounded input must result in a bounded output. In a fixed IIR filter this is not the case. Figure 3.4 shows a single weight IIR filter. The output y_t is generated by adding a delayed version of the output to the input x_t.

$$y_t = a_1 y_{t-1} + x_t \quad . \tag{3.1}$$

If the coefficient a_1 is of magnitude greater than 1, the filter output will "blow up". This is due to the output feedback inherent in IIR filtering making the filter unstable. When adapting IIR filters, one must make sure the filter weights remain in the stable region.

Another difficulty in IIR adaptive filtering is one of slow convergence. Most adaptive IIR filters require enormous amounts of data and consequently long waiting times until the filter converges. This is primarily due to the nature of the so called error surface which contains many flat areas. Because most adaptive algorithms are based on the method of steepest descent, flat areas on the error surface result in slow convergence. In the usual FIR filtering problem, this is not the case. The error surface of an FIR adaptive filter is a quadratic function of its weights and has only one minimum and usually does not possess flat spots.

The technique we have developed for adapting IIR filters involves minimizing an error which is somewhat different from the output error. The different

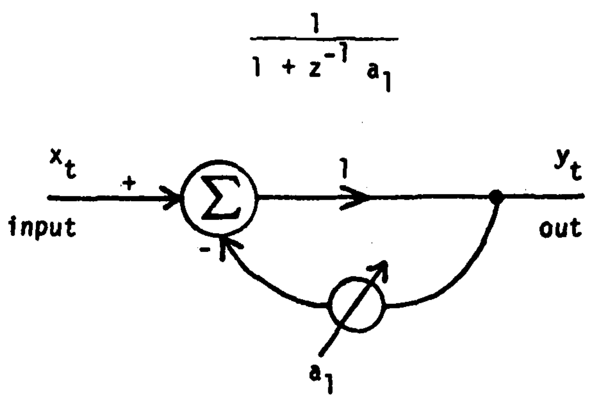
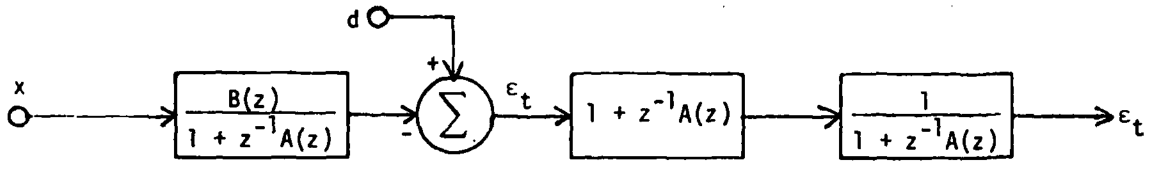


Figure 3.4. Single weight IIR filter.

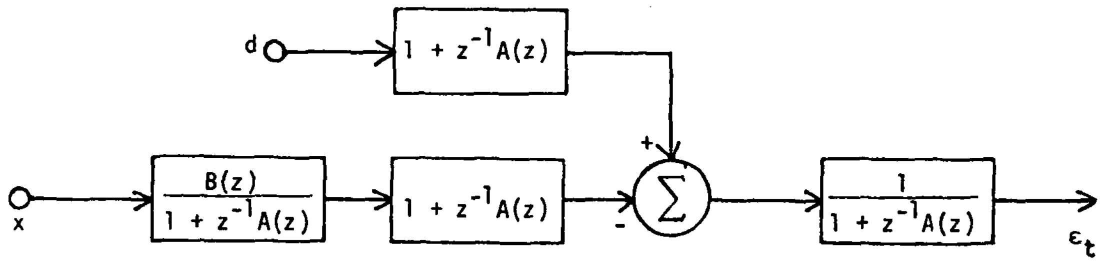
choice of error alleviates many of the problems just discussed. Consider a reformulation of the basic IIR adaptive filter shown in figure 3.3. In the path of the error, place an FIR and an IIR filter in cascade having transfer functions $1+z^{-1}A(z)$ and $\frac{1}{1+z^{-1}A(z)}$ respectively. This is shown in figure 3.5a. These two filters in cascade cancel and do not modify the overall structure. By pushing the first filter through the summation node, the structure shown in figure 3.5a is transformed into the identical structure shown in figure 3.5b. Now cancellation of the two cascaded filters results in the structure shown in figure 3.5c.

Notice the difference between the error e_t and the error e_t'; one is a filtered version of the other. The error e_t is called output error and the error e_t' is called "equation error". This terminology comes from control systems theory. It turns out that minimization of equation error is a much simpler task than minimization of output error. This is because the mean square of the output error is quadratic in the parameters of B(z), but non-quadratic function of the parameters of A(z). On the other hand, the mean square of equation error is a quadratic function of the parameters of both A(z) and B(z). But is it reasonable to minimize equation error rather than the natural output error? If the A(z) and B(z) polynomials have enough weights to make the equation error small, then it is likely that the output error will also be small. In fact, if the equation error is driven to zero, then the output error will also be zero. In this case, minimizing equation error is equivalent to minimizing output error.

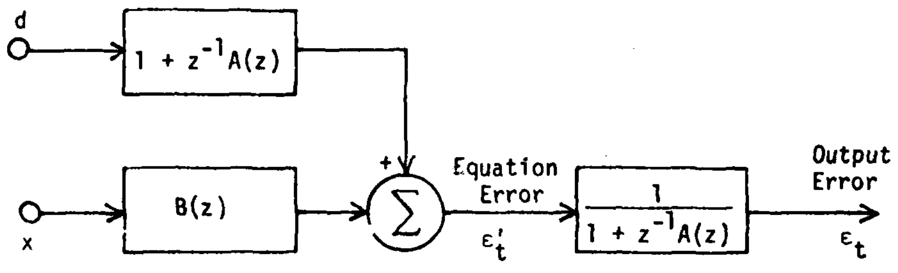
A filter based on minimizing equation error is shown in figure 3.6. The dashed lines indicate that the weights are to be copied into the output filter $\frac{1}{1+z^{-1}A(z)}$. The interesting feature about this structure is that it can be adapted using a simple FIR LMS algorithm. Consequently, it will have many of the nice robust characteristics that the FIR LMS algorithm has. The error sur-



(a)



(b)



(c)

Figure 3.5. IIR filter error reformulation.

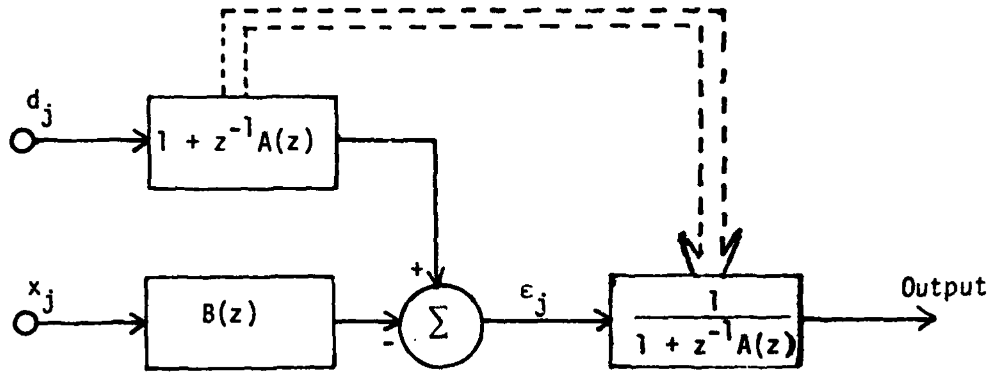


Figure 3.6. IIR adaptive filter minimizing equation error.

face will be a quadratic function of the weights; there will be no flat spots to slow convergence and no local minima to hinder global optimality.

One important detail has been neglected here. How can we be sure that the output filter $\frac{1}{1+z^{-1}A(z)}$ will be stable? The roots of the adaptive filter which is filtering the desired response could lie outside the stability region. As such, this would mean that $\frac{1}{1+z^{-1}A(z)}$ would be unstable. To overcome this problem, we have developed a minimum-phase constrained LMS algortihm. Minimum phase in this case means that all the roots of $1+z^{-1}A(z)$ lie within the unit circle (which is the desired stability region for $\frac{1}{1+z^{-1}A(z)}$) . If each update of the $1+z^{-1}A(z)$ polynomial is done so that the updated polynomial is kept minimum phase, then the output filter will remain stable.

The key to the new adaptive IIR filter is, in fact, the development of a minimum phase constrained LMS algorithm. This algorithm can be described as follows.

The minimum phase constrained LMS algorithm is a method of adapting polynomial coefficients to minimize the mean square error subject to the constraint that all roots of the polynomial lie either within the unit circle or within any circle of prescribed radius. As previously mentioned, the objective of this algorithm is to generate an invertible polynomial to be used in an adaptive IIR filter structure.

An update of the constrained polynomial is accomplished in two steps. The first step performs a conventional LMS update. The second step alters the updated polynomial in such a way that the resulting polynomial is minimum phase.

Step 1. Conventional LMS update:

$$A_{j+1} = A_j - \mu e_j X_j \tag{3.2}$$

where A_j is a vector containing the coefficients of the A(z) polynomial and e_j, X_j are defined as in figure 3.6.

Step 2. Minimum phase projection:
First check the $1+z^{-1}A(z)$ polynomial for minimum phase. If already minimum phase, this step is terminated. If A(z) is not minimum phase, then all roots of A(z) are shrunk radially towards the origin by a "shrinkage factor", ρ

$$A_{j+1}(z) \leftarrow A_{j+1}(\rho^{-1}z) \quad . \tag{3.3}$$

If ρ is chosen to be a positive number less than 1, then all of the roots of the $A_{j+1}(z)$ polynomial will be drawn radially inward. The shrinkage ratio, ρ, is reduced slowly in a series of small steps until A(z) passes the minimum phase test. This completes the adaptation cycle. When the next data vector is available, a new adapt cycle commences in the Step 1 mode.

Because the minimum phase test may need to be applied several times per adapt cycle, it is imperative that the test be done in a way that requires very little computation. An efficient test for minimum phase is one based on the lattice-form realization of FIR filters due to Itakura and Saito [17]. The polynomial $1+z^{-1}A(z)$ corresponds directly to a FIR transversal filter. An equivalent FIR lattice-form filter can be constructed from knowledge of the transversal filter weights. Both FIR filters will have the same impulse response. The polynomial $1+z^{-1}A(z)$ is minimum phase if and only if all of the weights (also known as reflection coefficients) of the equivalent lattice have magnitude less than one. The beauty of this test lies in the ease with which the transversal filter is transformed into an equivalent lattice filter. An algorithm for converting polynomial coefficients into the lattice filter reflection coefficients is described below. This algorithm lends itself readily to computer implementation, requiring memory space and computations proportional to the square of the filter order. A few years ago, we would not consider such an approach. Today, procedures of this type are with the realm of practicality.

The transversal filter polynomial to be tested can be represented by

$$1 + z^{-1}A(z) = 1 + a_1 z^{-1} + \cdots + a_m z^{-m} . \tag{3.4}$$

Define an nxm matrix Q and let Q(i,j) indicate the ith row and the jth column of this matrix. Initialize the bottom row of the Q matrix to the filter polynomial coefficients

```
For  $j = 1, 2, \ldots, m$
    Let  $Q(m,j) \leftarrow a_j$ .                                  (3.5)
```

Now implement the following recursion

```
For $i = m, m-1, \cdots, 1$
  For $j = 1, 2, \cdots, i-1$
    Let $Q(i-1,j) \leftarrow \dfrac{Q(i,j) - Q(i,i) \, x \, Q(i,i-j)}{1 - Q^2(i,i)}$ .        (3.6)
```

Let k_i be the ith reflection coefficient. These coefficients can now be read off the ith diagonal of the Q matrix

```
For  $j = 1, 2, \ldots, m$
    Let  $k_i \leftarrow Q(i,i)$ .                                  (3.7)
```

The filter is minimum phase if and only if all of the reflection coefficients are of magnitude less than one.

$$1+z^{-1}A(z) \text{ minimum phase} \quad \Leftrightarrow \quad k_i < 1 \text{ for } i = 1,2, \ldots, m . \tag{3.8}$$

This concludes the procedure for testing a polynomial for minimum phase and completes the overall description of the minimum phase constrained LMS algorithm.

The minimum phase LMS algorithm is currently being studied with the goal of developing convergence and optimality proofs. Although the algorithm is unproven mathematically, it has been tested extensively by computer simulation and has never failed to converge and find an optimal solution. It is a very important new development which will have applications in adaptive control, digital filter design, adaptive noise canceling, and adaptive antennas.

3.5 IIR Zahm Beamformer, a description

In this section, we show how to use the IIR adaptive filter to realize an IIR Zahm beamformer. Figure 3.7 shows a 3-element IIR Zahm beamformer configured to minimize output error. As we previously discussed, minimization of this output error is a difficult task. Instead we choose to reconfigure the system by a series of evolutionary steps shown in figure 3.8a-e, to minimize equation error.

First, we seperate the filters into all-zero and all-pole sections as shown in figure 3.8b. Next we establish a common denominator by inserting a filter $\frac{1}{(1+z^{-1}A_1(z))(1+z^{-1}A_2(z))}$ directly after the summer, and compensate by placing the inverse of this filter at the input to the summer. The introduction of these two filters has not changed the overall system, since they cancel each other's effects. Note that if the filter $1+z^{-1}A_i(z)$ is adapted in one place, all other copies $A_i(z)$ must receive the same update, which is indicated by the dashed lines. The new configuration of figure 3.8d behaves the same as that of figure 3.8a, since we have simply rearranged the system of figure 3.8a. A further simplification is to treat the cascaded filters $B_1(z)(1+z^{-1}A_2(z))$ and $B_2(z)(1+z^{-1}A_1(z))$ as single filters denoted by B_1' and B_2' respectively. The system of figure 3.8d will be adapted using the equation error e' rather than than the true output error e. This eliminates the IIR part of the beamformer from

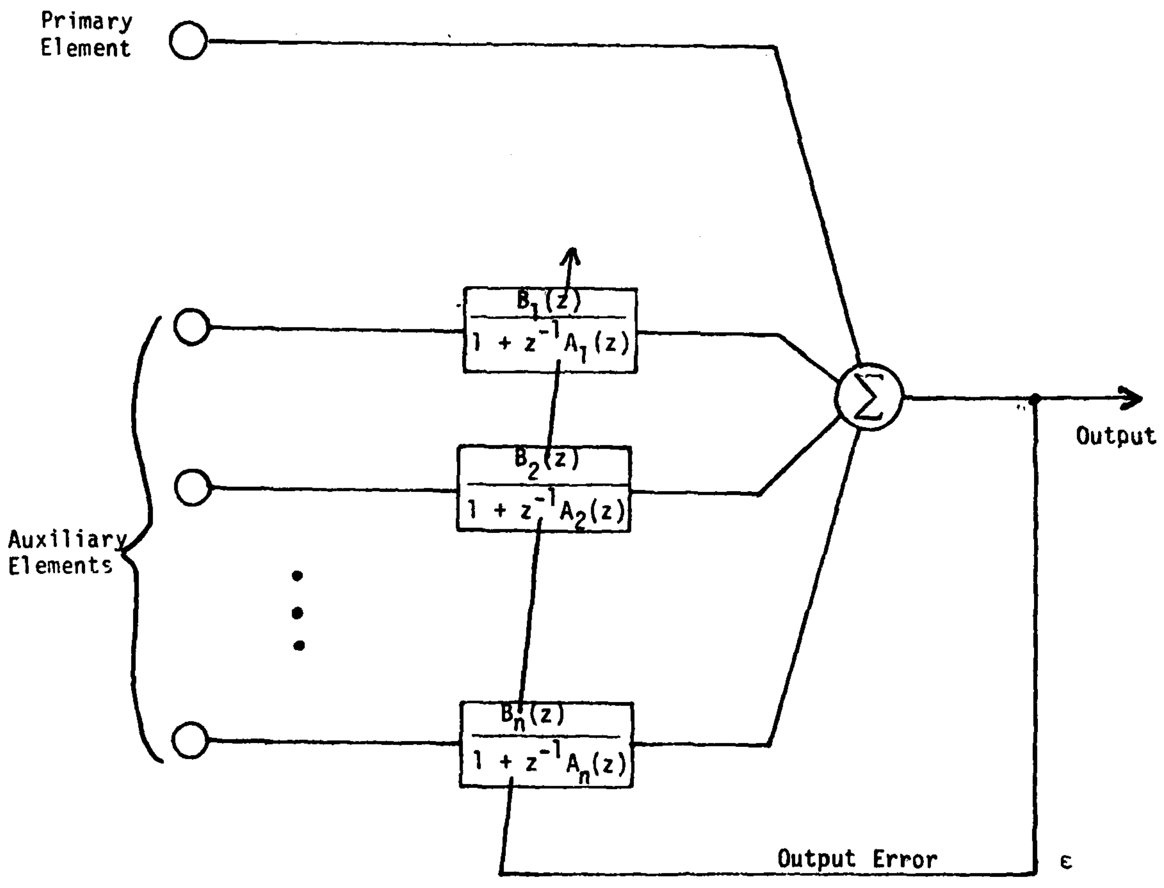

Primary Element
Auxiliary Elements
$\frac{B_1(z)}{1 + z^{-1}A_1(z)}$
$\frac{B_2(z)}{1 + z^{-1}A_2(z)}$
$\frac{B_n(z)}{1 + z^{-1}A_n(z)}$
Σ
Output
Output Error
ε

Figure 3.7. General IIR version of Zahm adaptive beamformer.

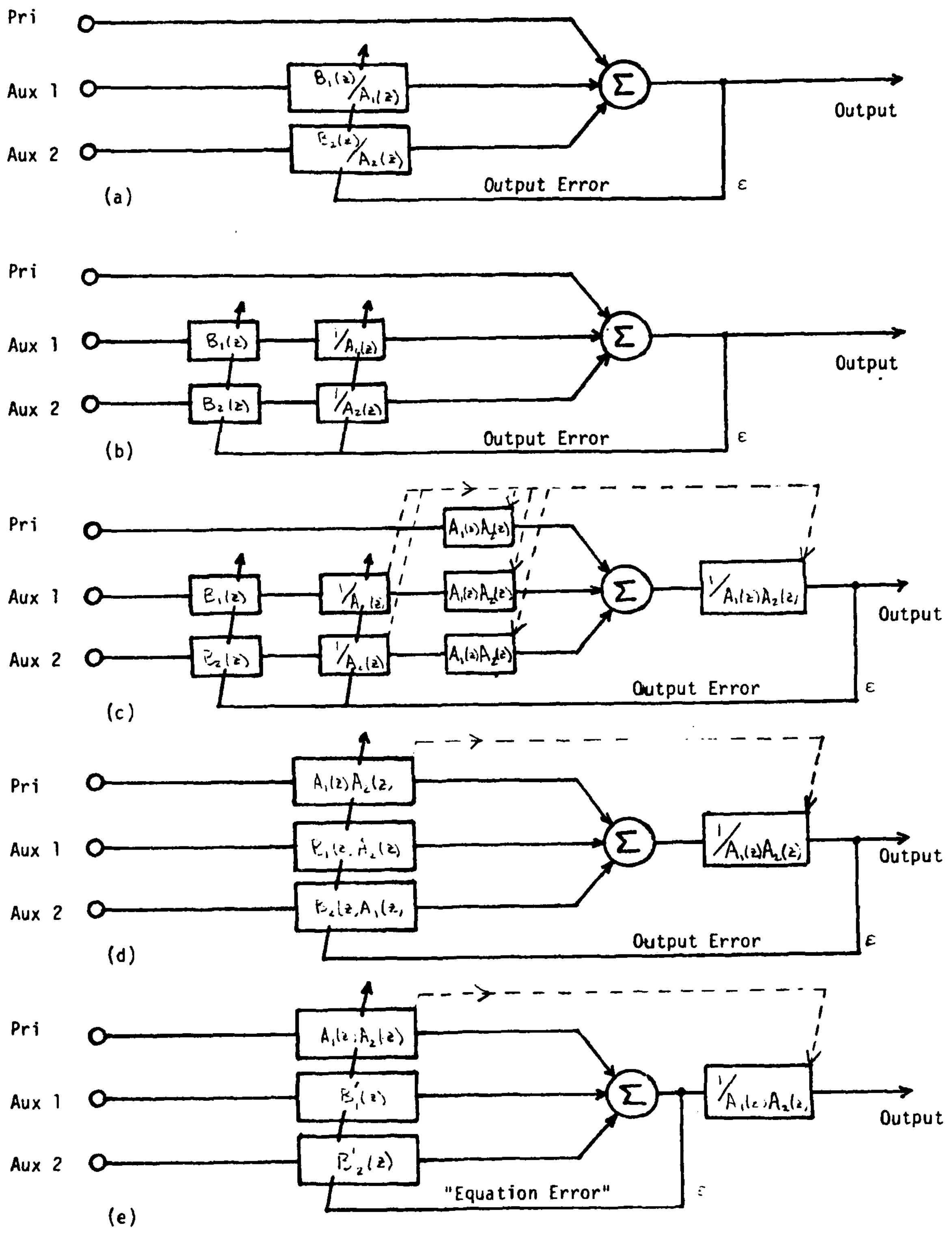


Figure 3.8. Simplification of three-element IIR beamformer.

the adaptation loop, so a simple algorithm for updating FIR filters, such as LMS, can be used. For simplicity we define the product of the two FIR filters as one single filter,

$$(1+z^{-1}A_1(z))(1+z^{-1}A_2(z)) = 1+z^{-1}A(z). \qquad (3.9)$$

The above result is easily generalized to a larger array with a greater number of auxilliary antenna elements and is shown in figure 3.9.

The weights, which are copied into the output filter $\frac{1}{1+z^{-1}A(z)}$, must not cause this filter to become unstable. To insure stability, we update the A(z) polynomial using the minimum phase LMS algorithm discussed in the previous section. The other polynomials are adapted with the conventional LMS algorithm.

In the next section, we will show simulation results using the IIR Zahm beamformer of figure 3.7. The $B_i(z)$ filters are adapted by LMS and the A(z) is adapted using minimum phase LMS.

3.6 IIR Zahm Beamformer, simulation results

In the following experiment, we compare the performance of a conventional Zahm beamformer to an IIR Zahm beamformer. To make a valid comparison, the number of degrees of freedom in both beamformers must be equal. Specifically, we will use a 4-element broadband Zahm beamformer with linear element placement and 4 taps per element. This corresponds to 12 degrees of freedom. The IIR Zahm beamformer will have 3 elements with the same linear placement and four taps per element corresponding to 11 degrees of freedom. The missing degree of freedom arises because of the fixed 1 in the $1+z^{-1}A(z)$ polynomial. Incident upon this array is a broadband jammer having a bandwidth equal to 10% of its center frequency. The desired signal is assumed to be of low power and, consequently has little effect on the converged beam pattern. The

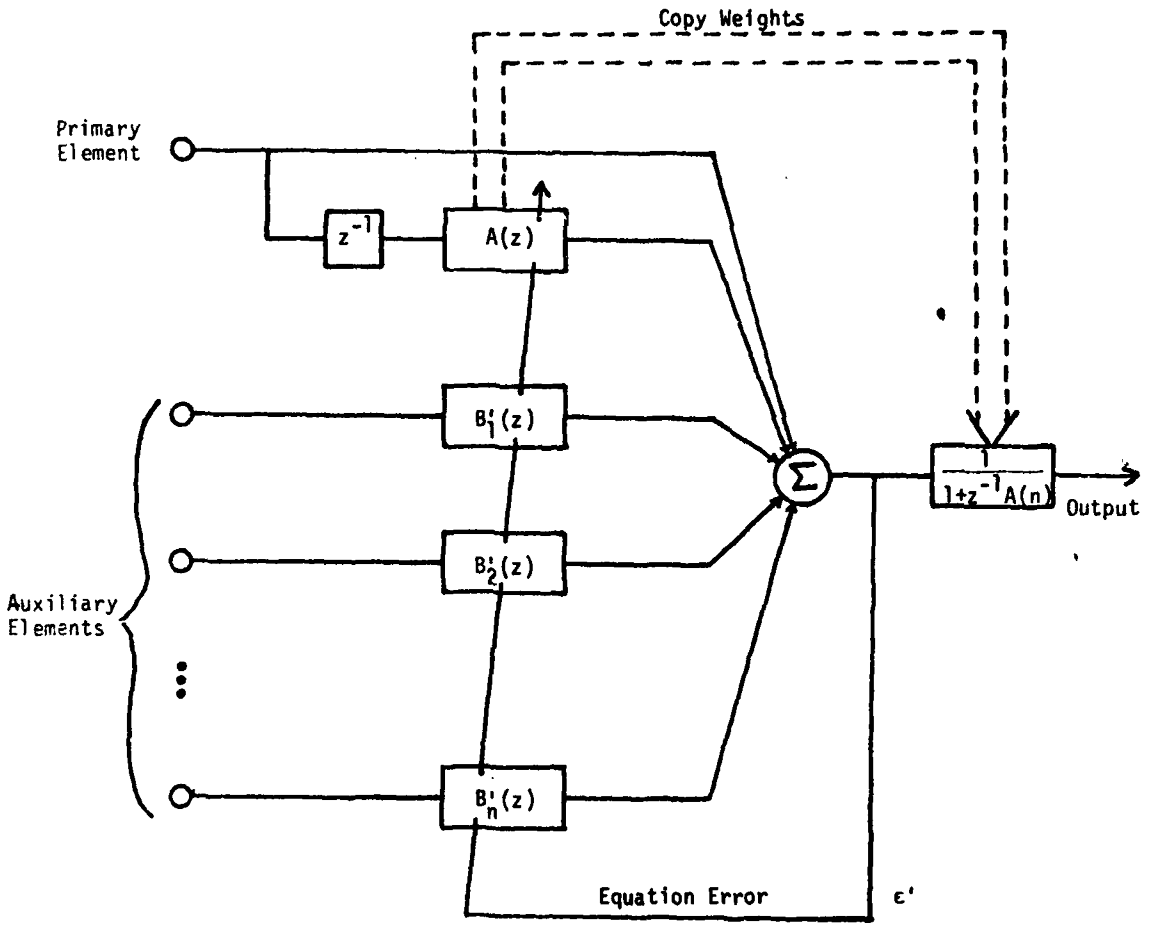
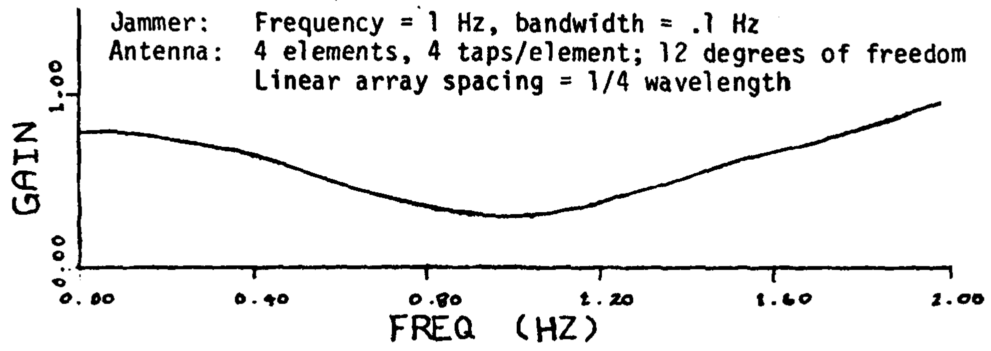


Figure 3.9. Simplified IIR version of Zahm adaptive beamformer.

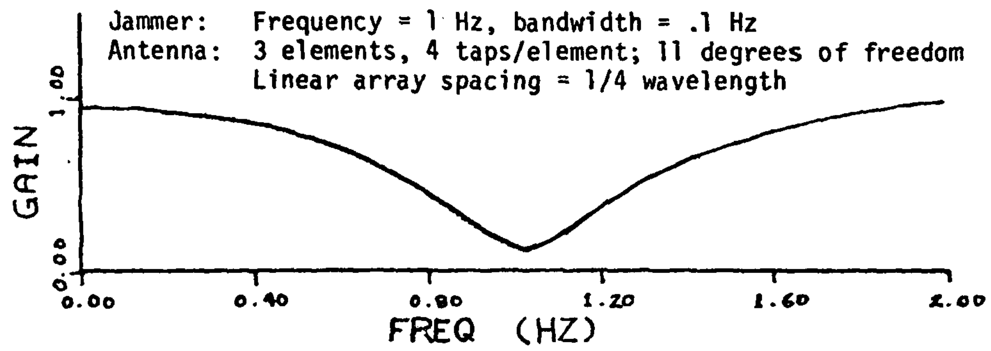
ideal behavior of this form of adaptive antenna would be to completely eliminate the jamming signal from the system output by forming a broadband null in the direction of the jammer. Unfortunately, perfect broadband nulls could only be obtained by implementing exact delays on all the antenna element circuits. With such delays the signals can be recombined in such a manner that perfect cancellation can occur. Physically, this would require an infinite range of adjustable delay. If two jammers were present, it would be theoretically impossible to form two perfect broadband nulls. Good broadband nulls can be obtained in practice using finely spaced delay line taps. As such, one can remove most of the broadband jamming power. Figure 3.10 shows the frequency reponse of the converged beamformer in the direction of the jammer for both the conventional Zahm array and the IIR Zahm array. Notice the improved notching performance of the IIR beamformer. It is able to remove a great deal more of the jammer power than the conventional beamformer. Although additional simulations need to be run, we feel that this simple example demonstrates the possible improvements which could be achieved using IIR beamforming.

3.7 Further Work

What we have seen thus far is probably just the "tip of an iceberg". Work is now proceeding to extend these results to an IIR version of a Frost beamformer. The Frost beamformer does not require the assumtion of a low-power desired signal, but does assume knowledge of the signal look-direction. Additional simulations and mathematical analysis must be done to verify the expected performance improvements under a wider range of signal, noise, and jammer conditions. Analytical expressions for optimal beam patterns are being derived.



(a) Conventional beamformer



(b) IIR beamformer

Figure 3.10. Comparison of frequency response in the direction of a broadband jammer for (a) conventional Zahm beamformer and (b) IIR Zahm beamformer. Notice the notch improvement obtained using the IIR beamformer.

APPENDICES

APPENDIX A

The matrix inversion lemma is given by

$$(A+BCD)^{-1} = A^{-1} - A^{-1}B[C^{-1}+DA^{-1}B]^{-1}DA^{-1} \tag{A.1}$$

choosing

$$A = \alpha R(t) = \left[\frac{1}{\alpha}P(t)\right]^{-1} \tag{A.2}$$

$$B = D^T = X(t+1) \tag{A.3}$$

$$C = 1\,. \tag{A.4}$$

gives

$$(\alpha R(t) + X(t+1)X^T(t+1))^{-1} = \frac{1}{\alpha}P(t) - \frac{1}{\alpha}P(t)X(t+1)$$

$$[1 + \frac{1}{\alpha}X^T(t+1)P(t)X(t+1)]^{-1}X^T(t+1)P(t)\frac{1}{\alpha}$$

$$= \frac{1}{\alpha}\left[P(t) - \frac{P(t)X(t+1)X^T(t+1)P(t)}{\alpha + X^T(t+1)P(t)X(t+1)}\right] \tag{A.5}$$

APPENDIX B

From the text, equation 2.20, the weight update is given as:

$$\underline{W}(t+1) = P(t+1)\Big(\sum_{i+1}^{t} \alpha\alpha_i(t)d(i)\underline{X}(i) + d(t+1)\underline{X}(t+1)\Big) \quad . \qquad \text{(B.1)}$$

Substituting Eq. (1a) gives

$$\begin{aligned}
\underline{W}(t+1) &= \frac{1}{\alpha}\Big(P(t) - \frac{P(t)\underline{X}(t+1)\underline{X}^T(t+1)P(t)}{\alpha + \underline{X}^T(t+1)P(t)\underline{X}(t+1)}\Big) \\
&\quad \Big(\sum_{i=1}^{t} \alpha\alpha_i(t)d(i)\underline{X}(i) + d(t+1)\underline{X}(t+1)\Big) \\
&= \underline{W}(t) + \frac{1}{\alpha}\Big(P(t)d(t+1)\underline{X}(t+1) - \frac{P(t)\underline{X}(t+1)\underline{X}(t+1)P(t)\alpha}{\alpha + \underline{X}^T(t+1)P(t)\underline{X}(t+1)} \\
&\quad \sum_{i=1}^{t} \alpha_i(t)d(i)\underline{X}(i) - \frac{P(t)\underline{X}(t+1)\underline{X}^T(t+1)P(t)}{\alpha + \underline{X}^T(t+1)P(t)\underline{X}(t+1)}\, d(t+1)\underline{X}(t+1)\Big) \\
&= \underline{W}(t) + \Big(\frac{1/2}{\alpha + \underline{X}^T(t+1)P(t)\underline{X}(t+1)}\Big)\Big(\alpha P(t)d(t+1)\underline{X}(t+1) + \\
&\quad d(t+1)P(t)\underline{X}(t+1)\underline{X}^T(t+1)P(t)\underline{X}(t+1) - P(t)\underline{X}(t+1)\underline{X}^T(t+1)P(t)d(t+1)\underline{X}(t+1) \\
&\quad - \alpha P(t)\underline{X}(t+1)\underline{X}^T(t+a)P(t) \sum_{i=1}^{t} \alpha_i(t)d(i)\underline{X}(i)\Big) \\
&= \underline{W}(t) + \Big(\frac{1/2}{\alpha + \underline{X}^T(t+1)P(t)\underline{X}(t+1))}P(t)\underline{X}(t+1)(\alpha d(t+1) - \alpha\underline{W}^T(t)\underline{X}(t+1] \qquad \text{(B.2)}
\end{aligned}$$

which gives the following recursive update for $\underline{W}(t)$

$$\begin{aligned}
\underline{W}(t+1) &= \underline{W}(t) + \frac{1}{\alpha + \underline{X}^T(t+1)P(t)\underline{X}(t+1)}P(t)\underline{X}(t+1)(d(t+1) \\
&\quad - \underline{W}^T(t)\underline{X}(t+1)) \quad . \qquad \text{(B.3)}
\end{aligned}$$

References

1. B. Widrow and M. E. Hoff, "Adaptive Switching Circuits," in 1960 WESCON Conv. Rec., pt. 4, pp. 96-140.

2. B. Widrow, P. Mantey, L. Griffiths, and B. Goode, "Adaptive Antenna Systems," Proc. IEEE, vol. 55, pp. 2143-2159, Dec. 1967.

3. B. Widrow et al., "Adaptive Noise Cancelling: Principles and Applications," Proc. IEEE, vol. 63, pp. 1692-1716, Dec. 1975.

4. B. Widrow, J. M. McCool, M. G. Larimore, and C. R. Johnson, Jr., "Stationary and Nonstationary Learning Characteristics of the LMS Adaptive Filter," Proc. IEEE, vol. 64, no. 8, pp. 1151-1162, August 1976.

5. B. Widrow and J. M. McCool, "A Comparison of Adaptive Algorithms Based on the Methods of Steepest Descent and Random Search," IEEE Trans. on Antennas and Propagation, vol. AP-24, no. 5, pp. 615-637, Sept. 1976.

6. P. E. Mantey and L. J. Griffiths, "Iterative Least-Squares Algorithms for Signal Extraction," Proceedings of the Second Hawaii International Conference of System Sciences, pp. 767-770, 1968.

7. L. J. Griffiths, "An Adaptive Lattice Structure for Noise Cancelling Applications," IEEE Int. Conf. on Accoustics, Speech and Signal Processing, Tulsa, Ok. pp. 87-90, Apr. 1978.

8. L. E. Brennan and I. S. Reed, "Convergence Rate in Adaptive Arrays," Technology Service Corp. Report No. TSC-PD-177-4, Jan. 13, 1978.

9. C. A. Baird, "Recursive Algorithms for Adaptive Arrays," Final Report, Contract No. F30602-72-C-0499, Rome Air Development Center, Sept. 1973, 778947/2GI

10. A. Oppenheim and R. Schafer, Digital Signal Processing, Prentice Hall Inc., New Jersey, 1975.

11. B. Widrow, P. Titchner, W. Newman and R. Gooch, "Research on Algorithms for Adaptive Antenna Arrays," Monthly Letter Report #3, Rome Air Development Center, contract no. F30602-80-C-0046, May 1980. Stanford University, Stanford, CA May 1980.

12. B. Widrow, K. Duvall, R. Gooch, and W. Newman, "Jamming of Adaptive Arrays by Signal Cancellation: The Phenomenon and Two Remedies," Symposium on Adaptive Antennas, Colorado Springs, CO, July 8-10, 1980.

13. A. Papoulis, Probability, Random Variables and Stochastic Processes, McGraw-Hill Book Co., New York, 1965.

14. R. W. Sittler, "Lectures on Sampled Data Systems Analysis," Massachusetts Institute of Technology, Lincoln Laboratory, pp. 99-101, August 1957.

15. P. O. Feintuch, "An Adaptive Recursive LMS Filter," Proc. IEEE, vol. 64, no. 11, pp. 1622-1624, Nov. 1976.

16. S. A. White, "An Adaptive Recursive Digital Filter,"in Proc. 9th Asilomar Conf. on Circuits, Systems, and Computers, p. 21, Nov. 1975.

17. F. Itakura and S. Saito, "Digital Filtering Techniques for Speech Analysis and Synthesis," Proc. 7th Int. Cong. Accoust. Budapest, paper 25-C-1, pp. 261-264, 1971.

AD-A106 684 STANFORD UNIV CA F/G 9/5
RESEARCH ON ALGORITHMS FOR ADAPTIVE ANTENNA ARRAYS.(U)
AUG 81 B WIDROW, W NEWMAN, R GOOCH, K DUVALL F30602-80-C-0046
UNCLASSIFIED RADC-TR-81-206 NL

2 of 2

END
DATE
FILMED
12 81
DTIC

D

FI

2 -